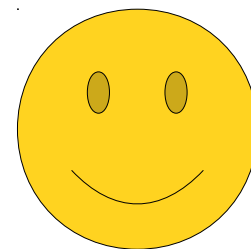
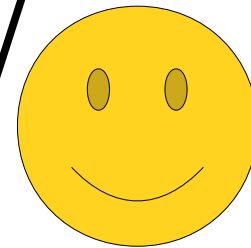


# Assignment 0: Using the Debugger

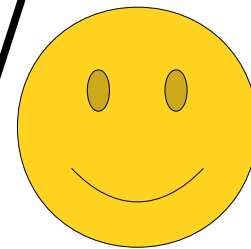
Hi everybody!



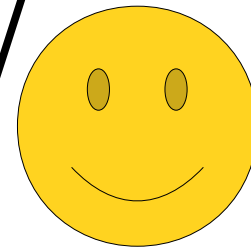
As part of Assignment 0, we'd like you to get a little bit of practice using the debugger in Qt Creator.



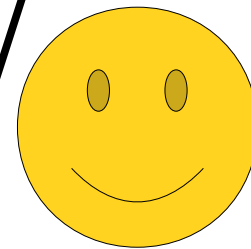
The debugger is a tool you can use to help see what your program is doing as you run it.



It's really useful for helping find errors in your programs, and the more practice you get with it, the easier it'll be to correct mistakes in the programs you write.



Think of this guide as a little tutorial walkthrough to help give you a sense of how to use the debugger and how to make sense of what you're seeing.



To start things off, open up the Name Hash program you ran in Part One of this assignment. Scroll down to the nameHash function so that you can see the entire function in your window.



```
40 * of the
41 *
42 * For tho
43 * treats
44 * It then
45 * F_p, whe
46 * some smaller prime numbers (16908799 and 127),
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is close to
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Qt IDE interface showing the 'name\_hash.cpp' file. The interface includes a top menu bar (File, Edit, Build, Debug, Analyze, Tools, Window, Help), a left sidebar with tool icons, a central editor window displaying the C++ code, and a bottom status bar with tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages. The system tray in the top right corner shows the time as 10:24 AM and the page number as 1 of 1.

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```
40 * of the input and produces a number.  
41 *  
42 * For those of you who are more mathematically inclined, this function  
43 * treats each character in the input name as a number between 0 and 128.  
44 * It then uses them as coefficients in a polynomial over the finite field  
45 *  $F_p$  where  $p$  is a large prime number, and evaluates that polynomial at  
this for CS106B,
```

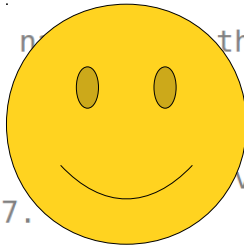
Move your mouse cursor so that it's in the space right before the line number for line 66.

Now, click the mouse!

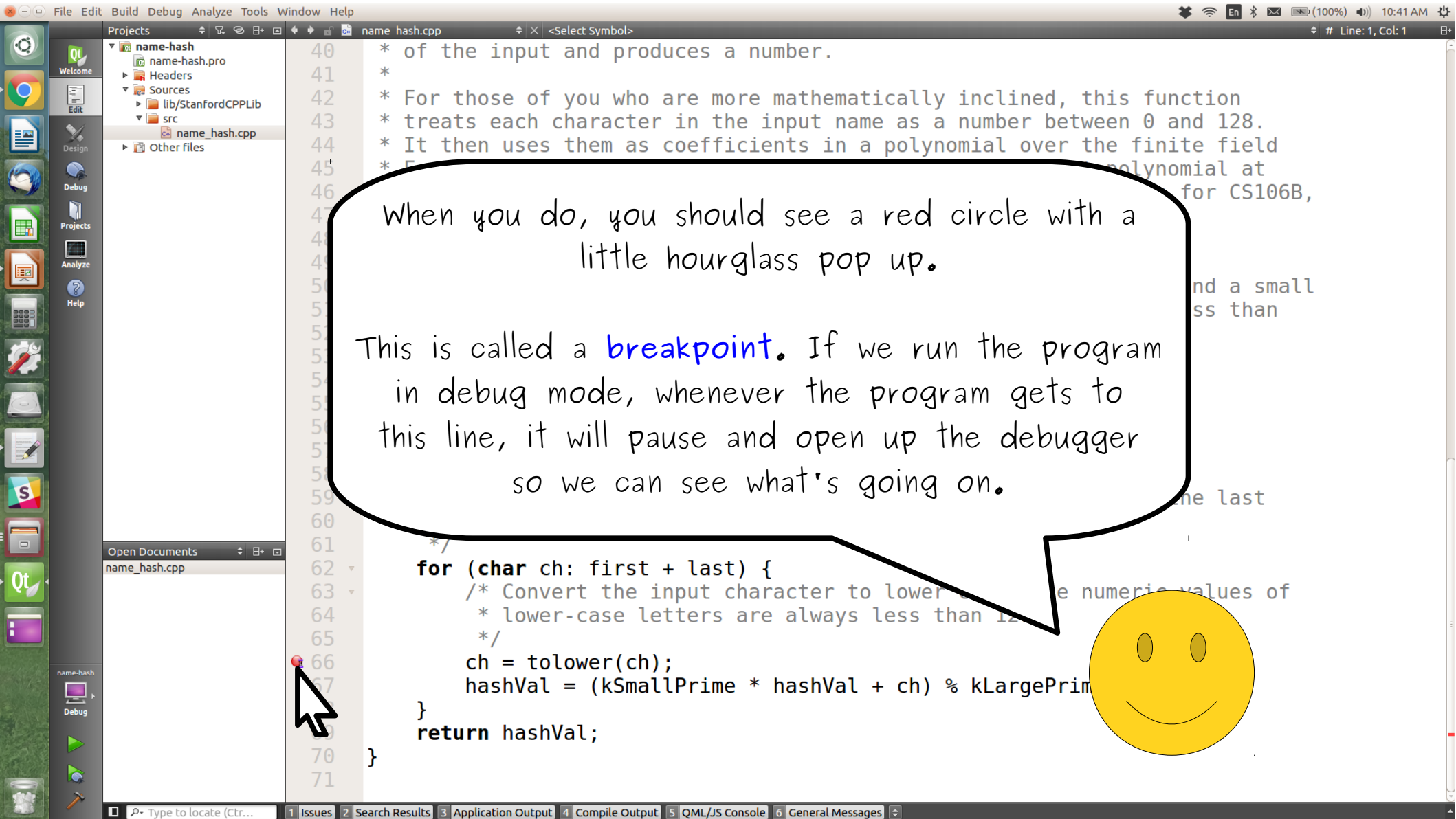
Open Documents

- name\_hash.cpp

```
56  
57 int hashVal = 0;  
58  
59 /* Iterate across all the characters in the name from first to the last  
60 * name, updating the hash at each step.  
61 */  
62 for (char ch: first + last) {  
63     /* Convert the input character to lower case. The values of  
64     * lower-case letters are always less than 127.  
65     */  
66     ch = tolower(ch);  
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
68 }  
69 return hashVal;  
70 }  
71
```

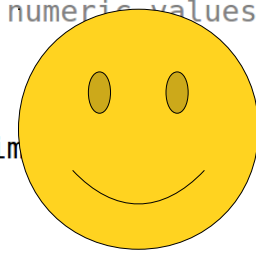






When you do, you should see a red circle with a little hourglass pop up.

This is called a **breakpoint**. If we run the program in debug mode, whenever the program gets to this line, it will pause and open up the debugger so we can see what's going on.



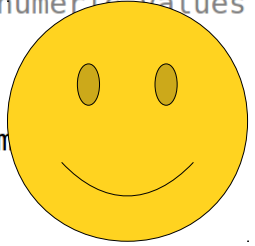
Qt IDE interface showing a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The 'Debug' icon is highlighted with a mouse cursor.

```

40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS106B,
47 * but we thought it might be fun!)
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a small
51     * prime, both of which are less than
52     * the number of characters in the last
53     * character of the input string.
54     * The large prime is kLargePrime and the small prime is kSmallPrime.
55     * The numeric values of
56     * the characters are converted to lower case.
57     * Lower case letters are always less than 128.
58     * The numeric values of
59     * the characters are converted to lower case.
60     * Lower case letters are always less than 128.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64         * lower-case letters are always less than 128.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71

```

Now, we're going to run this program in debug mode. To do so, click on the "run in debug mode" button in the bottom-right corner of the screen. It's the one just below the regular green "run" button. When you do...



... you should see something like this! Notice that a bunch of extra panels popped up in Qt Creator. We'll talk about what each of these windows mean in a second.

The screenshot shows the Qt Creator IDE interface. The main editor displays a C++ file named `name_hash.cpp` with the following code:

```
41
42
43 * treats each character in the input name as a coefficient in a polynomial
44 * It then uses them as coefficients in a polynomial of the form
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!)
48 */
49 int nameHash(const std::string& name) {
50     /*
51     *
52     *
53     *
54     *
55     *
56     *
57     *
58     *
59     */
60     int hashVal = 0;
61     for (char ch : name) {
62         /*
63         *
64         *
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
```

A console window is open in the foreground, displaying the prompt "What is your first name? |".

The debugger window at the bottom shows the following table:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

The status bar at the bottom indicates the current location: `1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages`.



In the meantime, type in the first name Ada and hit enter, as shown here.



Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays code for a hash function. A console window is open, showing the program's output: 'What is your first name? Ada' and 'What is your last name?'. The console window also shows the Qt logo and 'Application started'. The bottom status bar shows '1 Issues', '2 Search Results', '3 Application Output', '4 Compile Output', '5 QML/JS Console', and '6 General Messages'. A 'Build' button is visible in the bottom right corner.

```
41 int nameHash(const QString &name) {
42     // This function
43     // treats each character in the input name as a coefficient in a polynomial
44     // It then uses them as coefficients in a polynomial of the form
45     // F_p, where p is a large prime number, and evaluates that polynomial at
46     // some smaller prime number q. (You aren't expected to know this for CS
47     // but we thought it might be fun!)
48     */
49     int nameHash(const QString &name) {
50         /*
51          * What is your first name? Ada
52          * What is your last name?
53          *
54          *
55          *
56          *
57          *
58          *
59          */
60         /*
61          *
62          *
63          *
64          *
65          *
66          *
67         */
68         ch = tolower(ch);
69         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
70     }
71 }
```

Now, type in "Lovelace" as a last name, but  
don't hit enter yet!



Qt IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name\_hash.cpp". A console window is open, showing the program's output: "What is your first name? Ada" and "What is your last name? Lovelace". The console window is partially overlapping the code editor. The Qt IDE interface includes a sidebar with various tool icons (Welcome, Edit, Design, Debug, Projects, Analyze, Help) and a bottom status bar with tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.

```
41
42
43 * treats each character in the input name as a coefficient in a polynomial
44 * It then uses them as coefficients in a polynomial of the form
45 *  $F_p$ , where  $p$  is a large prime number, and evaluates that polynomial at
46 * some smaller prime number  $q$ . (You aren't expected to know this for CS
47 * but we thought it might be fun!)
48 */
49 int nameHash(const std::string& name) {
50     /*
51     * What is your first name? Ada
52     * What is your last name? Lovelace
53     *
54     *
55     *
56     *
57     *
58     *
59     */
60     /*
61     *
62     *
63     *
64     *
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
```

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

As soon as you hit enter, a bunch of things are going to pop up in Qt Creator. Don't panic! It's normal.

The screenshot shows the Qt Creator IDE interface. On the left is a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main editor displays a C++ file named `name_hash.cpp` with the following code:

```
41 int nameHash(const std::string& name) {  
42     // This function  
43     * treats each character in the input name as a coefficient in a polynomial  
44     * It then uses them as coefficients in a polynomial of degree n-1 for the finite field  
45     * F_p, where p is a large prime number, and evaluates that polynomial at  
46     * some smaller prime number q. (You aren't expected to know this for CS  
47     * but we thought it might be fun!)  
48     */  
49     int hashVal = 0;  
50     for (int i = 0; i < name.length(); i++) {  
51         char ch = name[i];  
52         // What is your first name? Ada  
53         // What is your last name? Lovelace  
54         // *  
55         // *  
56         // *  
57         int chVal = ch - 'a';  
58         // *  
59         // *  
60         // *  
61         // *  
62         // *  
63         // *  
64         // *  
65         // *  
66         ch = tolower(ch);  
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
68     }  
69     return hashVal;  
70 }
```

A console window is open in the foreground, showing the program's output:

```
File Edit Options Help  
What is your first name? Ada  
What is your last name? Lovelace
```

A yellow smiley face is drawn over the console window. At the bottom, the debugger shows a thread table with the following data:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

The status bar at the bottom shows various toolbars and tabs: Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.

With that said, hit enter,  
and watch the magic happen!

```
41 int nameHash(const QString &name) {
42     // This function
43     // treats each character in the input name as a coefficient in a polynomial
44     // It then uses them as coefficients in a polynomial over the finite field
45     // F_p, where p is a large prime number, and evaluates that polynomial at
46     // some smaller prime number q. (You aren't expected to know this for CS
47     // but we thought it might be fun!)
48     */
49     int nameHash(const QString &name) {
50         /*
51          * What is your first name? Ada
52          * What is your last name? Lovelace
53          *
54          *
55          *
56          *
57          *
58          *
59          *
60          *
61          *
62          *
63          *
64          *
65          *
66          *
67          */
68     }
69     ch = tolower(ch);
70     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
71 }
```

Qt IDE interface showing the 'name-hash' project. The console window displays the following text:

```
What is your first name? Ada
What is your last name? Lovelace
```

A yellow smiley face is overlaid on the console window.

The background code shows a function `nameHash` that calculates a hash value for a name. The function uses a polynomial over a finite field  $F_p$  and evaluates it at a smaller prime number  $q$ . The code includes comments explaining the process and a note that the user is not expected to know this for CS.

The Qt IDE interface includes a sidebar with various tools and a bottom status bar with tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.



Shazam! We're back in Qt Creator, and there's tons of values showing up everywhere.



```
47  */
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less than
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the last
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



There's a lot going on right here. Let's see what's happening.



```
47  */
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  *  $2^{31} - kLargePrime - 1$ .
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

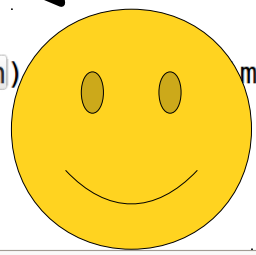
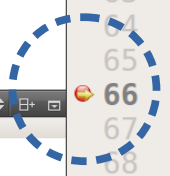
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50      /* This hashing scheme needs two prime numbers, a large prime and a
51      * prime. These numbers were chosen because their product is less th
52      * 202711141. Prime Numbers:
53      * 202711141 = 127 * 16908799
54      */
55      int hashVal = 0;
56      for (char ch: first + last) {
57          /* Convert the input character to lower case. The numeric values
58          * of lower-case letters are always less than 127.
59          */
60          ch = tolower(ch);
61          hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
62      }
63      return hashVal;
64  }
65
66
67
68
69
70
71

```

First, notice that our red breakpoint now has a yellow arrow in it.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

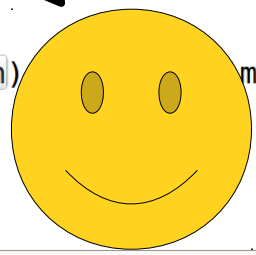
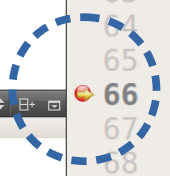
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50      /* This hashing scheme needs two prime numbers, a large prime and a
51      * prime. These numbers were chosen because their product is less th
52      * 202714311. Primes 127 and 16908799 were chosen because their
53      * product is less than 2^31.
54      */
55      int hashVal = 0;
56      for (char ch: first + last) {
57          /* Convert the input character to lower case. The numeric values
58          * of lower-case letters are always less than 127.
59          */
60          ch = tolower(ch);
61          hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
62      }
63      return hashVal;
64  }
65
66
67
68
69
70
71

```

This yellow arrow indicates where in the program we are right now. The program stopped running at this line because we hit that breakpoint you set earlier.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

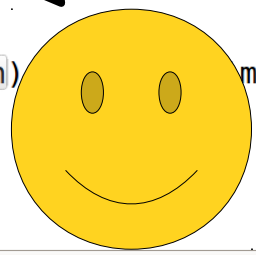
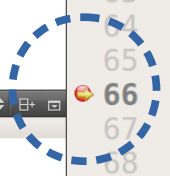
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50      /* This hashing scheme needs two prime numbers, a large prime and a
51      * prime. These numbers were chosen because their product is less th
52      * 202714311. Primes 127 and 16908799 were chosen because their
53      * product is less than 2^31.
54      */
55      int hashVal = 0;
56      for (char ch: first + last) {
57          /* Convert the input character to lower case. The numeric values
58          * of lower-case letters are always less than 127.
59          */
60          ch = tolower(ch);
61          hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
62      }
63      return hashVal;
64  }
65  }
66  }
67  }
68  }
69  }
70  }
71  }

```

Whenever you pop up the debugger, it's good to figure out exactly where you are in the program that you're running, so you'll get into the habit of checking for this yellow arrow.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

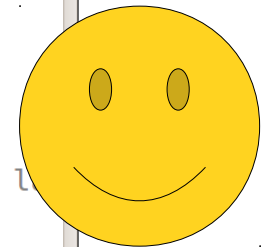
- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
      - src
        - name\_hash.cpp
  - Other files

Open Documents

- name\_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch =
67         hashV
68     }
69     return ha
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



Next, let's take a look at this panel.  
This is called the **call stack**.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

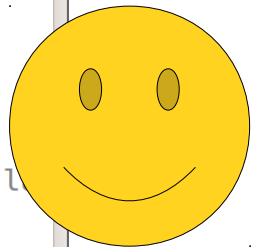
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

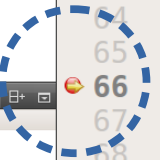
- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch = hashVal
67     }
68
69     return ha
70 }
71 }
```



Right now, we know we're in the nameHash function, because our helpful friend the Yellow Arrow tells us exactly what line we're on!



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

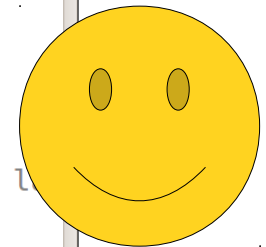
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- name\_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch = hashVal
67     }
68     return ha
69 }
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



However, the yellow arrow can't tell us exactly how we got to this part of the program. What part of the program actually called nameHash?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

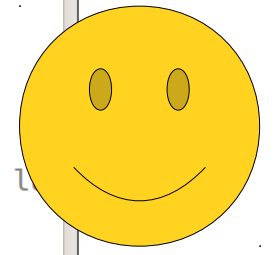
- name\_hash.cpp

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the l
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The num
64  * lower
65  */
66  ch =
67  hashV
68  }
69  return ha
70  }
71

```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



The call stack can tell us exactly that!

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								



Projects

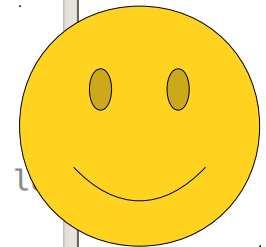
- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
      - src
        - name\_hash.cpp
  - Other files

Open Documents

- name\_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch = hashV
67     }
68     return ha
69 }
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



Notice that the call stack lists a series of different functions in order. Here, it has nameHash (where we are now) at the top, and right below that is Main.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...)

- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Projects

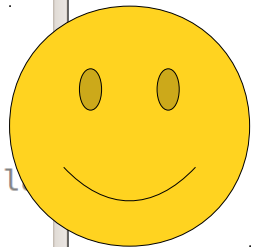
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the l
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The nu
64  * lower
65  */
66  ch =
67  hashV
68  }
69  return ha
70  }
71

```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



Go and double-click the call to Main on Level 1.  
When you do...

Open Documents

- name\_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	25								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

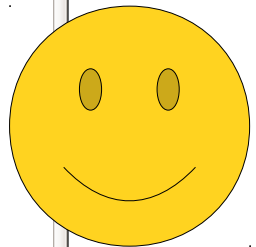
Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual implementation of the nameHash function. It
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where n is

```



... you'll end up over here!

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	766504679	int

Open Documents

name\_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name\_hash.cpp". A yellow smiley face is drawn next to the code, with a speech bubble pointing to line 31. The speech bubble contains the text: "Notice that the highlighted line here includes a call to the nameHash function. This the part of the code that actually called nameHash, which is how we got to the line with the breakpoint!". The IDE also shows a variable watch window on the right and a debug console at the bottom.

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual implementation of the nameHash function.
38  * to talk to the user and get the input.
39  * the meaning of the input.
40  * of the input.
41  *
42  * For those who are interested, the function
43  * treats each character as a number.
44  * It then uses a formula to calculate the hash value.
45  * For example, the hash value for "John" is 15.
```

Variable Watch Window:

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	766504679	int

Debug Console:

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

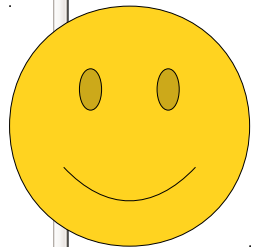
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Bottom Panel: 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name\_hash.cpp". The code includes "console.h" and "simpio.h", and defines a "nameHash" function and a "main" function. A breakpoint is set at line 31, where the program has paused. A yellow smiley face is drawn next to the code, with a speech bubble containing the text: "Generally speaking, you can use the call stack as a way to see which function calls got us to the point where the program paused at the breakpoint!". The bottom of the IDE shows the "Threads" window, which displays the call stack with the following data:

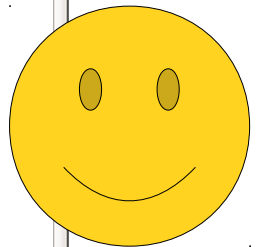
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Generally speaking, you can use the call stack as a way to see which function calls got us to the point where the program paused at the breakpoint!



Projects sidebar showing a tree view for 'name-hash' project, including folders for Headers, Sources, lib/StanfordCPPLib, and src, with the file 'name\_hash.cpp' selected.

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual implementation of the nameHash function.
38  * to talk to the user and get the input.
39  * the meaning of the input.
40  * of the input.
41  *
42  * For those who are interested,
43  * treats each character as a
44  * It then uses a simple
45  * F n where n is the number of characters in the string.
```



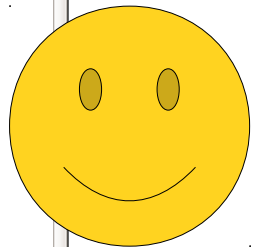
You might notice that there's some more stuff in the call stack beyond just main and nameHash. What are those?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, Open Documents, and Debug.

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual implementation of the nameHash function.
38  * to talk to the user and get the input.
39  * the meaning of the input.
40  * of the input.
41  *
42  * For those who are interested,
43  * treats each character as a
44  * It then uses a simple
45  * F n where n is the number of characters in the string.
```



Let's find out! Double-click on the line marked "Main" on Level 2. When you do...

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startUpMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

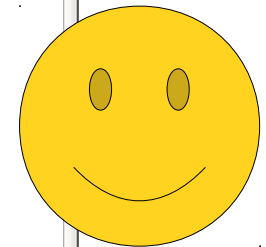
Qt  
Welcome  
Edit  
Design  
Debug  
Projects  
Analyze  
Help

Open Documents  
main.cpp  
name\_hash.cpp

name-hash  
name-hash.pro  
Headers  
Sources  
lib/StanfordCPPLib  
src  
name\_hash.cpp  
Other files

name-hash  
Debug

```
1  /* ... */  
17  
18 #include <iostream>  
19  
20 #ifndef SPL_AUTOGRADER_MODE  
21 int Main(int, char* /*argv*/[]) {  
22     extern int Main();  
23     return Main();  
24 }  
25 #endif // SPL_AUTOGRADER_MODE  
26
```



... you'll end up with something that looks like this.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

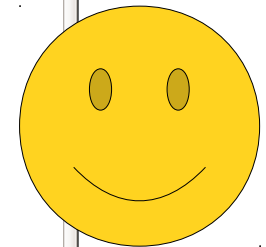


Qt  
Welcome  
Edit  
Design  
Debug  
Projects  
Analyze  
Help

Open Documents  
main.cpp  
name\_hash.cpp

name-hash  
name-hash.pro  
Headers  
Sources  
lib/StanfordCPPLib  
src  
name\_hash.cpp  
Other files

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



Yikes! This looks hairy and scary! What happened?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

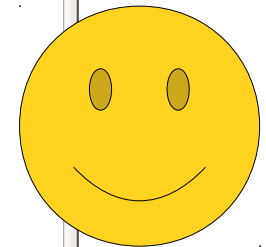
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt  
Welcome  
Edit  
Design  
Debug  
Projects  
Analyze  
Help

Open Documents  
main.cpp  
name\_hash.cpp

name-hash  
name-hash.pro  
Headers  
Sources  
lib/StanfordCPPLib  
src  
name\_hash.cpp  
Other files

```
1  /* ... */  
17  
18 #include <iostream>  
19  
20 #ifndef SPL_AUTOGRADER_MODE  
21 int Main(int, char* /*argv*/[]) {  
22     extern int Main();  
23     return Main();  
24 }  
25 #endif // SPL_AUTOGRADER_MODE  
26
```



Whenever you start up a program in CS106B, there's a little bit of code that we automatically call for you, which does things like setting up the console.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

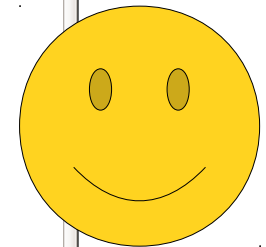
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt  
Welcome  
Edit  
Design  
Debug  
Projects  
Analyze  
Help

Open Documents  
main.cpp  
name\_hash.cpp

```
1  /* ... */  
17  
18 #include <iostream>  
19  
20 #ifndef SPL_AUTOGRADER_MODE  
21 int Main(int, char* /*argv*/[]) {  
22     extern int Main();  
23     return Main();  
24 }  
25 #endif // SPL_AUTOGRADER_MODE  
26
```



This code will show up in the call stack below your actual program.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

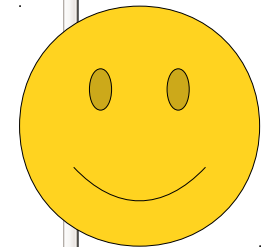
Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



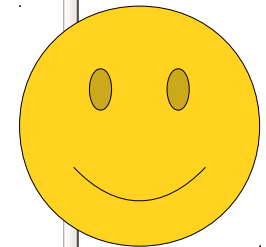
You shouldn't need to dig around this deep in the call stack, and if you do, it should probably be a message telling you to back up a bit back to code that you actually wrote.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



so let's jump back to the code that we actually wrote.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

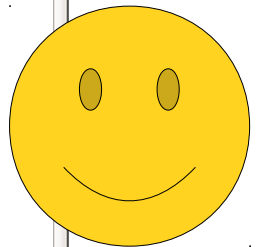
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
        - src
          - name\_hash.cpp
  - Other files

```
1  /* ... */
17
18 #include <iostream>
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```



To do that, double-click on Level 0, the call to nameHash. When you do...

Open Documents

- main.cpp
- name\_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

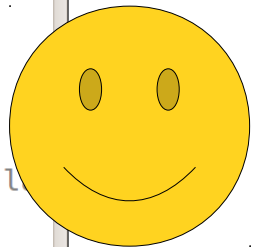
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
      - src
        - name\_hash.cpp
  - Other files

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the l
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The num
64         * lower
65         */
66         ch =
67         hashV
68     }
69     return ha
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



You'll be teleported back to safety!

Open Documents

- main.cpp
- name\_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
      - src
        - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the string 'first'
60     * and calculate the hash value.
61     */
62     for (const char& ch : first)
63     {
64         /*
65         *
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Let's quickly recap what we've seen so far.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								





Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

name-hash

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50      /* This hashing scheme needs two prime numbers, a large prime and a
51      * prime. These numbers were chosen because their product is less th
52      * 2^31 - kLargePrime - 1.
53      */
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63      /* Convert the input character to lower case. The numeri
64      * lower-case letters are always less than 127.
65      */
66      ch = tolower(ch);
67      hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

Once the breakpoint is reached, it will pull up all sorts of useful information.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								





Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeri
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

Now, let's see how we can read the values of the variables in this function.



Open Documents

- main.cpp
- name\_hash.cpp

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Look up at this panel over here.



```
45
46
47 * but we
48 */
49
50 nameHash(string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
__for_begin	@0x7fffffff030
__for_end	@0x7fffffff040
__for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

This window lets you take a look at all the values of the local variables that are in scope right now.



```
45
46
47 * but we
48 */
49
50 nameHash(string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
▶ __for_begin	@0x7fffffff030
▶ __for_end	@0x7fffffff040
▶ __for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Open Documents

- main.cpp
- name\_hash.cpp

name-hash

Debug

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Depending on what OS you're using, these might be in a different order, and there might be some weird-looking ones in there in addition to nicer ones like `ch` and `hashVal`.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
__for_begin	@0x7fffffff030
__for_end	@0x7fffffff040
__for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								



If we ignore the weird-looking ones, we can see some nice, familiar names.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value
__for_begin	@0x7fffffff030
__for_end	@0x7fffffff040
__for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

For example, here you can see the values of kLargePrime and kSmallPrime, which match the values they were declared with.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing the project structure on the left, the code editor in the center, and the debug console at the bottom. The project structure includes 'name-hash' with subfolders for 'Headers', 'Sources', and 'lib/StanfordCPPLib'. The code editor shows the implementation of the 'nameHash' function. The debug console shows the execution flow, including the 'nameHash' function call and the 'Main' function.

We can also see that, at this point, hashVal is still zero.



```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Type to locate (Ctrl...

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

As we walk through the program one step at a time, we'll see these values change.



```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

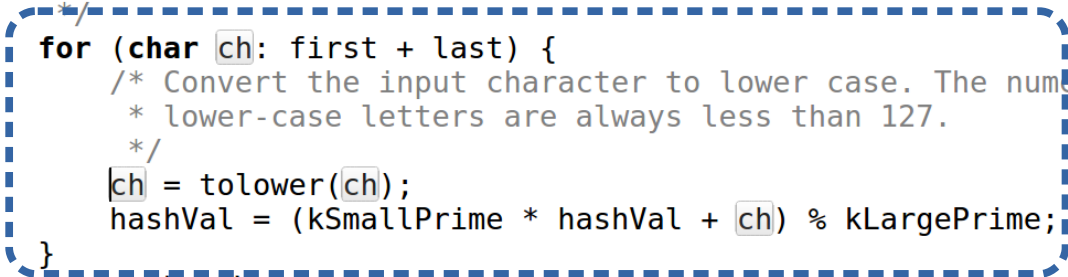
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing the project structure on the left, the source code in the center, and the debug console at the bottom. The project structure includes 'name-hash' with subfolders for 'name-hash.pro', 'Headers', 'Sources', 'lib/StanfordCPPLib', and 'src'. The source code is 'name\_hash.cpp'. The debug console shows the current thread and breakpoint information.

Now, let's take a look at this for loop.



```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

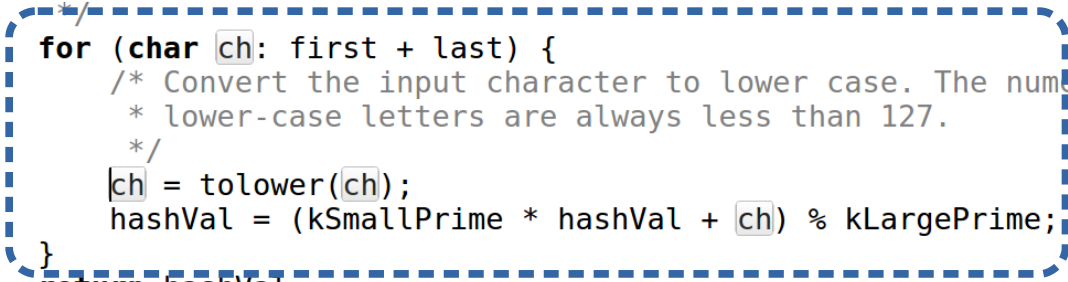
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

This loop is a **range-based for loop**. It says "for each character in the string first + last, do something with that character."



```
45
46
47 * but we
48 */
49 nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

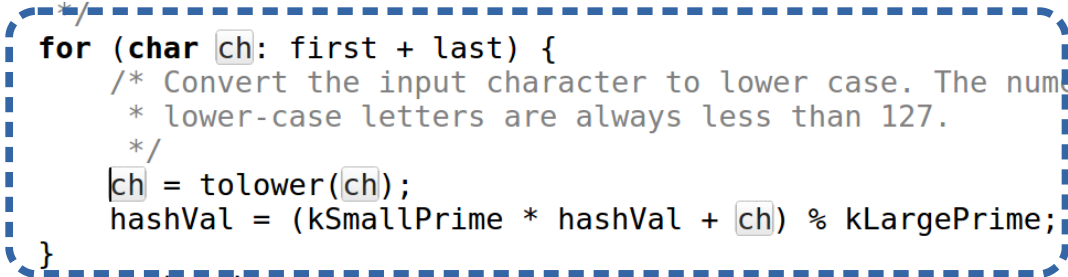
Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing the project structure on the left, the code editor in the center, and the debug console at the bottom. The project structure includes 'name-hash' with subfolders for 'Headers', 'Sources', and 'src'. The code editor shows the 'name\_hash.cpp' file. The debug console shows the current thread and breakpoint information.

Remember (from a while back) that we entered the name **Ada Lovelace**.



```
45
46
47 * but we
48 */
49
50 nameHash(const string first, string last){
51     /* This hashing scheme needs two prime numbers, a large prime and a
52     * small prime. These numbers were chosen because their product is less th
53     *  $2^{31} - kLargePrime - 1$ .
54     */
55     static const int kLargePrime = 16908799;
56     static const int kSmallPrime = 127;
57
58     int hashVal = 0;
59
60     /* Iterate across all the characters in the first name, then the las
61     * name, updating the hash at each step.
62     */
63     for (char ch: first + last) {
64         /* Convert the input character to lower case. The numeric values
65         * lower-case letters are always less than 127.
66         */
67         ch = tolower(ch);
68         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69     }
70     return hashVal;
71 }
```



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages





So now we know where we are (line 66), how we got there (main called nameHash), and the values in the program at this point.



```
45
46
47 * but we
48 */
49 nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Now, let's do something really cool - we're going to run this program one line at a time, watching what happens at each step!

The screenshot shows the Qt Creator IDE interface. The main window displays a C++ source file named 'name\_hash.cpp' with the following code:

```
45
46
47 * but we
48 */
49 nameHash(const string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * small prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

The IDE is in a debug state, with a breakpoint set at line 66. The 'Threads' panel at the bottom shows the current execution stack:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

The 'Open Documents' panel on the left shows 'main.cpp' and 'name\_hash.cpp'. The 'Debug' panel at the bottom shows the current thread is 'name-hash' and it is stopped at breakpoint 1 (1) in thread 1.

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	ch
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Qt IDE interface showing a C++ code editor with a stack trace at the bottom. The code is for a name hashing function. A red box highlights the stack trace, and a yellow smiley face is next to a callout box.

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Stack Trace:

Level	Function	File	Line
0	nameHash	name_hash...	66
1	main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Callout text: Right above the stack trace, you'll see there are some small button icons.

Qt IDE interface showing a C++ code editor with a nameHash function. The code includes comments about prime numbers and a loop that iterates over characters in two strings, calculating a hash value. A red box highlights the Qt debugger toolbar, and a callout bubble explains its functions.

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```


Qt Debugger toolbar buttons: Run, Step Over, Step Into, Step Out, Break, Continue, Stop.

Callout bubble text: "These buttons let you resume the program, stop the program, walk through it one line at a time, etc."

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	65 'A'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Qt IDE interface showing a C++ code editor with a nameHash function. The code includes comments about prime numbers and a for loop that iterates over characters. A red box highlights a button in the debug toolbar. A yellow smiley face and a speech bubble are overlaid on the image.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```



Move your mouse so that you're hovering over the button that's third from the left. If you hover over it, it should say "step over."

Level	Action	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt IDE interface showing a C++ code editor with a function `nameHash`. The code calculates a hash value for a string based on two prime numbers, `kLargePrime` and `kSmallPrime`. The function iterates over each character, converts it to lowercase, and updates the hash value using the formula: `hashVal = (kSmallPrime * hashVal + ch) % kLargePrime`.

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

The Qt IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main editor shows the code with line numbers. A red box highlights the 'Step Over' button in the debug toolbar. A yellow smiley face is positioned to the right of the code. A speech bubble contains the text: "Once you're confident that you're on the 'step Over' button - and not the 'step Into' or 'step Out' buttons - go and click it! When you do..."

Level	Action	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt IDE interface showing a C++ code editor with the following code:

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

The code is annotated with a yellow smiley face and a speech bubble pointing to line 67:

...your window should look something like this.

The IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main window shows the code editor, a variable declaration window, and a threads window.

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Level	Function	File	Line
0	nameHash	name_hash...	67
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name\_hash.cpp'. The code defines a 'nameHash' function that takes two strings, 'first' and 'last', and returns an integer hash value. The function uses two prime numbers, 'kLargePrime' (16908799) and 'kSmallPrime' (127), to calculate the hash. The hash is calculated by iterating over all characters in both strings, converting them to lowercase, and applying a formula:  $hashVal = (kSmallPrime * hashVal + ch) \% kLargePrime$ . The code is as follows:

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71 }
```

The Qt IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The 'Open Documents' panel shows 'main.cpp' and 'name\_hash.cpp'. The 'Threads' panel at the bottom shows the execution stack: 0 nameHash (line 67), 1 Main (line 31), 2 Main (line 23), 3 startupMain (line 2208), and 4 main (line 27). A yellow smiley face is drawn on the right side of the code editor, and a speech bubble points to the hash calculation line (67) with the text: "Okay! A few things have changed. Let's see what's going on."



Qt IDE interface showing a C++ project named "name-hash" with source files "name\_hash.cpp" and "main.cpp". The main editor displays the implementation of a name hashing function.

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71

```

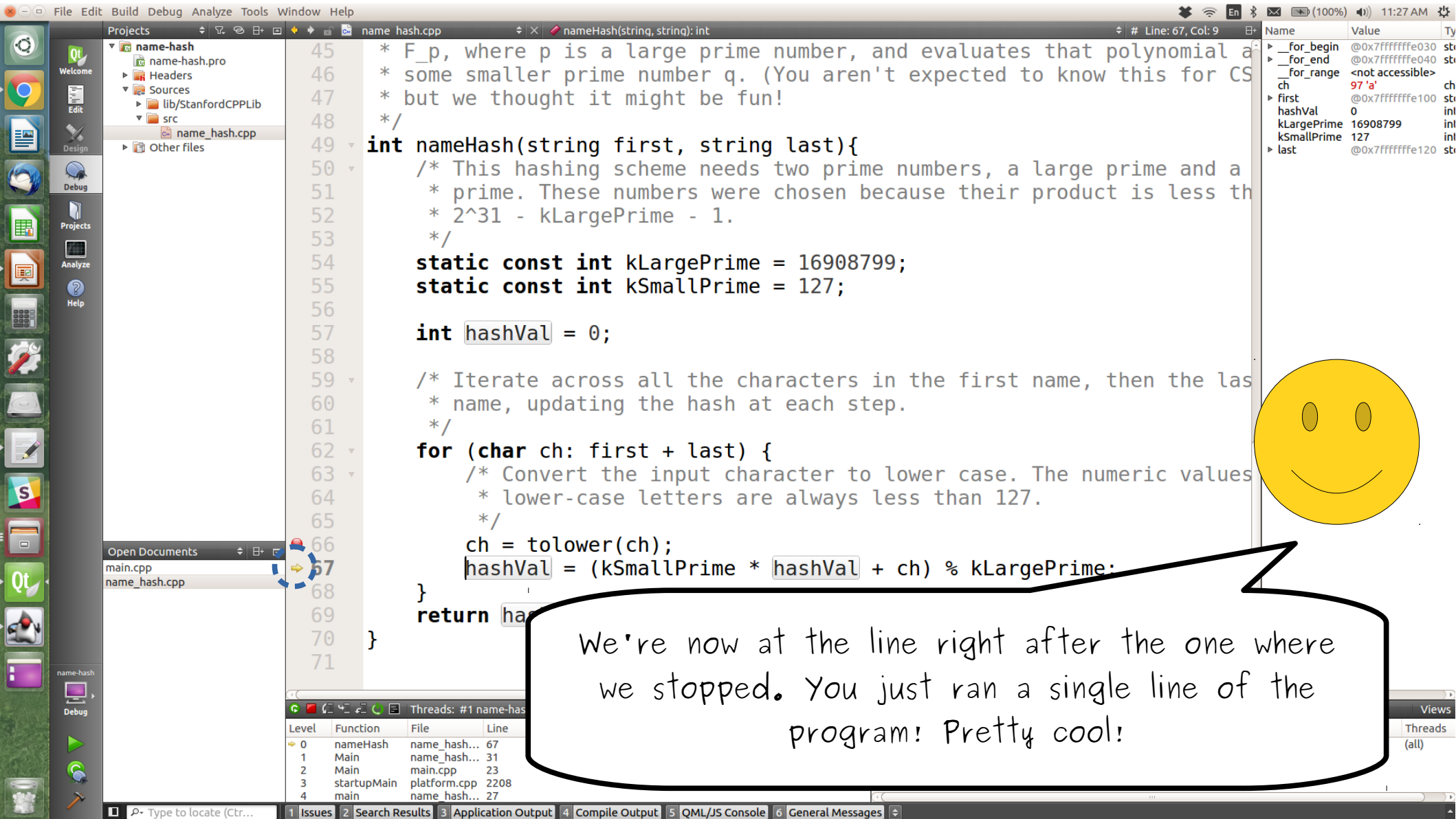
The Qt debugger shows the execution stack with the following details:

Level	Function	File	Line
0	nameHash	name_hash...	67
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

A yellow smiley face icon is present on the right side of the editor.

First, notice that our helpful yellow Arrow friend is now pointing at line 67.


The status bar at the bottom shows various panels: Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.



Qt IDE interface showing a C++ code editor with the following code:

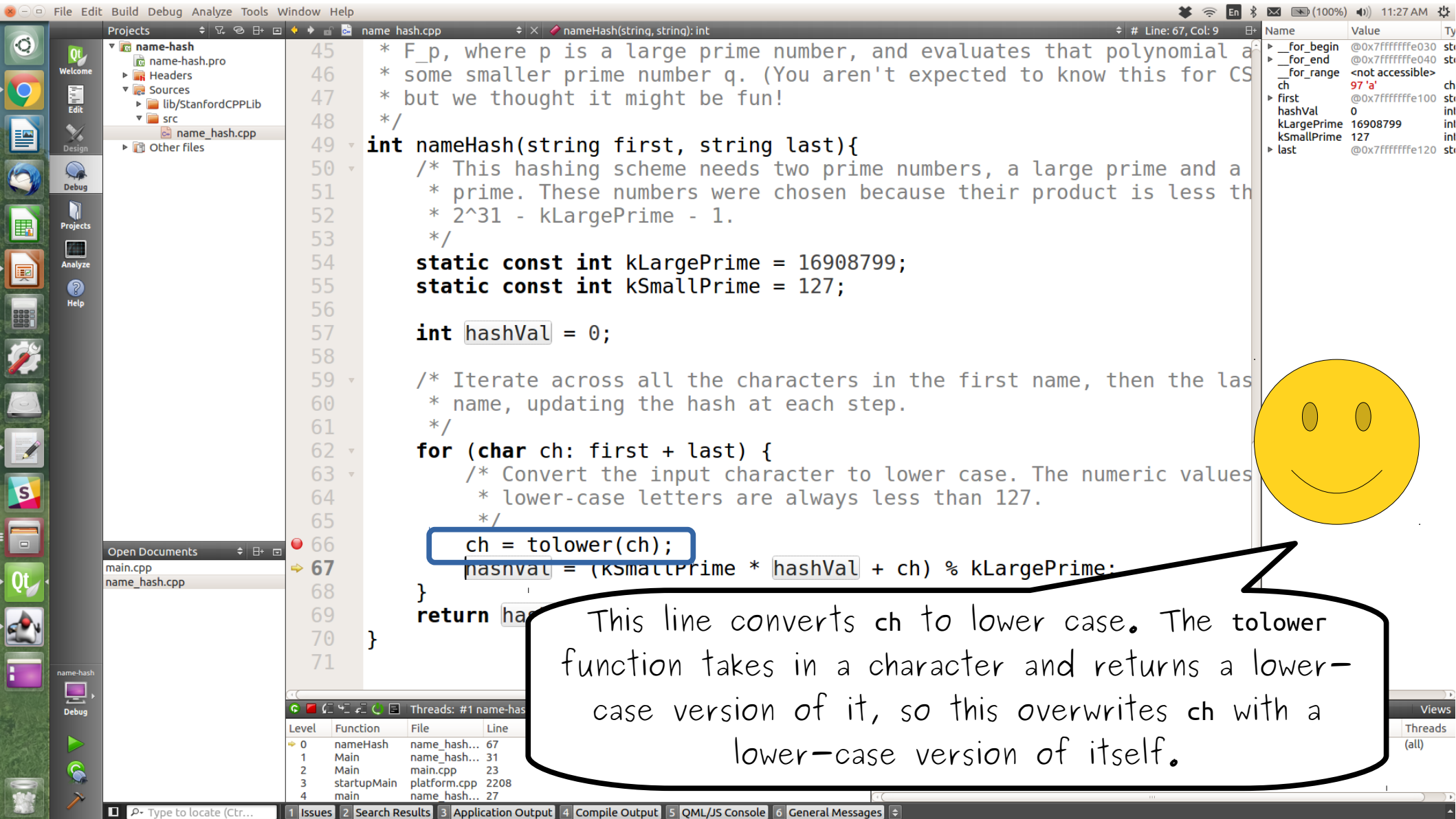
```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric values
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

The code is annotated with a yellow smiley face and a speech bubble pointing to line 67:



so what did that line of code do?

The IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The bottom status bar shows tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.



This line converts ch to lower case. The tolower function takes in a character and returns a lower-case version of it, so this overwrites ch with a lower-case version of itself.

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

name-hash

```

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

```

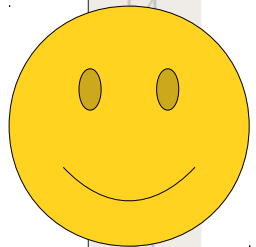
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

```

You can actually see this by looking at the values panel over on the side!



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

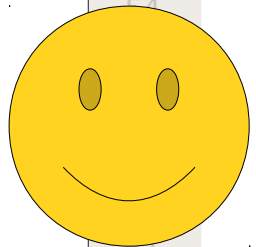
Open Documents

- main.cpp
- name\_hash.cpp

name-hash

Debug

Notice that the value associated with ch has changed from 'A' to 'a' - it's now in lower-case!



```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

```

Name	Value	Type
for_begin	@0x7fffffff030	str
for_end	@0x7fffffff040	str
for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								



Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

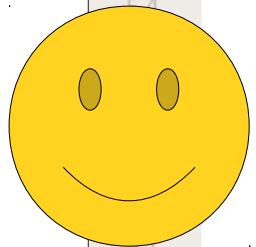
```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

```
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
}
```

If you'll notice, this value is in red while all the other values are in black.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, Open Documents, and Debug.

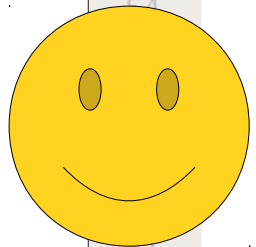
```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

This indicates that the value here has changed since the previous step. This is a really useful way to keep track of what's changing as you run the program.

```
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								



Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

name-hash

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  *
47  *
48  *
49  *
50  *
51  *
52  *
53  *
54  *
55  *
56  *
57  *
58  *
59  *
60  *
61  *
62  *
63  *
64  *
65  *
66  *
67  *
68  *
69  *
70  *
71  *

```

static const int kLargePrime = 16908799;  
static const int kSmallPrime = 127;

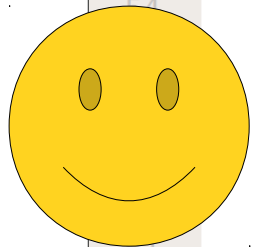
int hashVal = 0;

/\* Iterate across all the characters in the first name, then the last name, updating the hash at each step.  
\*/

for (char ch: first + last) {  
 /\* Convert the input character to lower case. The numeric values of lower-case letters are always less than 127.  
 \*/  
 ch = tolower(ch);  
 hashVal = (kSmallPrime \* hashVal + ch) % kLargePrime;  
}

return hashVal;

Now, let's take a look at line 67, where we are right now.



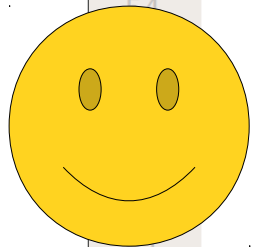
Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects sidebar showing 'name-hash' project structure: name-hash.pro, Headers, Sources (lib/StanfordCPPLib, src), name\_hash.cpp, Other files.

Open Documents sidebar showing 'main.cpp' and 'name\_hash.cpp'.



Not gonna lie, this is a pretty dense line of code. It performs some weird sort of mathematical calculation on a bunch of different values.

45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71

```
* F_p, where p is a large prime number, and evaluates that polynomial a
* for CS
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;
int hashVal = 0;
/* Iterate across all the characters in the first name, then the las
* name, updating the hash at each step.
*/
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
    * lower-case letters are always less than 127.
    */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

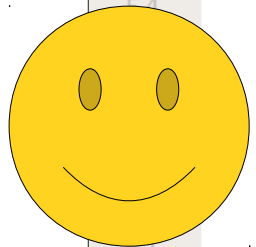
Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects sidebar showing 'name-hash' project structure: name-hash.pro, Headers, Sources, lib/StanfordCPPLib, src, name\_hash.cpp, Other files.

Open Documents sidebar showing 'main.cpp' and 'name\_hash.cpp'.

Fundamentally, though, it's just computing some weird function of some values and stashing it into hashVal.



45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71

```
* F_p, where p is a large prime number, and evaluates that polynomial a
* for CS
static const int kLargePrime = 16908799,
static const int kSmallPrime = 127;
int hashVal = 0;
/* Iterate across all the characters in the first name, then the las
* name, updating the hash at each step.
*/
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
    * lower-case letters are always less than 127.
    */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

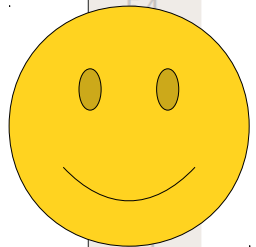
name-hash

```

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54
55
56 static const int kLargePrime = 16908799,
57 static const int kSmallPrime = 127;
58
59 int hashVal = 0;
60
61 /* Iterate across all the characters in the first name, then the las
62 * name, updating the hash at each step.
63 */
64 for (char ch: first + last) {
65     /* Convert the input character to lower case. The numeric values
66     * lower-case letters are always less than 127.
67     */
68     ch = tolower(ch);
69     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
70 }
71 return hashVal;

```

Let's go run that line of code and see what happens!



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing a C++ code editor with the following code:

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

The IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. A 'Projects' pane on the left shows the file structure for 'name-hash'. A 'Debug' console at the bottom shows a stack trace with a red box around the 'Step Over' button. A yellow smiley face is on the right side of the screen.

Hover over the "step Over" button, confirm that the button you're clicking really is "step Over," and click it! When you do...

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the implementation of a name hashing function. The code includes comments explaining the polynomial hashing scheme and defines two prime constants: `kLargePrime = 16908799` and `kSmallPrime = 127`. The function `nameHash` iterates over characters from two strings, converting them to lowercase and updating a hash value using the formula `hashVal = (kSmallPrime * hashVal + ch) % kLargePrime`. A yellow smiley face is drawn over the code, with a speech bubble pointing to the calculation line that says "... you'll end up with something like this!". The right sidebar shows a memory dump with values like `@0x7fffffff030` and `97 'a'`. The bottom status bar shows tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric val
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71 }
```

... you'll end up with something like this!

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Level	Function	File	Line
0	nameHash	name_hash...	62
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

Qt IDE interface showing a C++ project named 'name-hash'. The main editor window displays the source code for 'name\_hash.cpp', which implements a name hashing function. The code includes comments explaining the hashing scheme and defines two prime constants, kLargePrime and kSmallPrime. A yellow smiley face is drawn over the code, and a speech bubble points to the function call in the main function.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric val
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71 }
```

Let's see what's changed.

Level	Function	File	Line
0	nameHash	name_hash...	62
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27





Qt  
Welcome  
Edit  
Design  
Debug  
Projects  
Analyze  
Help

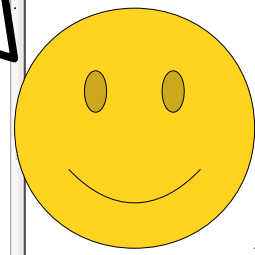
Open Documents  
main.cpp  
name\_hash.cpp

name-hash  
name-hash.pro  
Headers  
Sources  
lib/StanfordCPPLib  
src  
name\_hash.cpp  
Other files

name-hash  
Debug

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the string.
60     * name, up to last.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric value
64         * of lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71
```

Second, notice that we're back up at the top of the for loop, since that's where the yellow arrow is pointing. We ended up back here because this is the next line that gets executed.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

name-hash

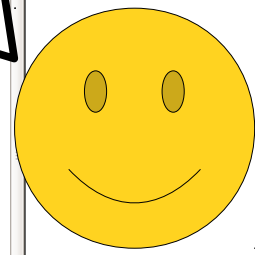
Debug

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the range [first, last) and compute
60  * the hash value.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71

```

We just single-stepped through a single iteration of that loop! Pretty cool!



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	97 'a'	ch
first	@0x7fffffff100	str
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Projects

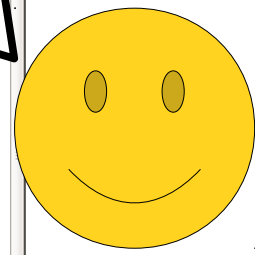
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the range [first, last)
60  * name,
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71

```

Let's go do it again!



Open Documents

- main.cpp
- name\_hash.cpp

Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

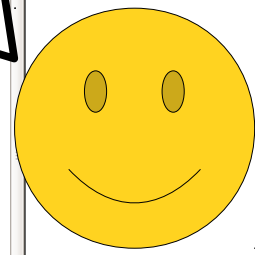
name-hash

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the range [first, last)
60  * name,
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71

```

Again, move your mouse over the Step Over button (and make sure it says "Step Over" and not something else!), then click it.



Name	Value	Type
__for_begin	@0x7fffffff030	std::string
__for_end	@0x7fffffff040	std::string
__for_range	<not accessible>	std::string
ch	97 'a'	char
first	@0x7fffffff100	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	std::string

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name\_hash.cpp', which implements a polynomial hash function. The code is as follows:

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate over the characters in the range [first, last).
60     * name,
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric value of
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

The variable `ch` is highlighted in the code, and a speech bubble points to it with the text: "Now we're here! Notice that `ch` now has the value 'd', which is the second letter of the name Ada." A yellow smiley face is drawn next to the speech bubble.

The right-hand pane shows the Qt Variable Inspector, which displays the current state of variables in the scope. The variable `ch` is highlighted, showing its value as `100 'd'`.

Name	Value	Type
<code>_for_begin</code>	<code>@0x7fffffff030</code>	<code>str</code>
<code>_for_end</code>	<code>@0x7fffffff040</code>	<code>str</code>
<code>ch</code>	<code>100 'd'</code>	<code>ch</code>
<code>first</code>	<code>@0x7fffffff100</code>	<code>str</code>
<code>hashVal</code>	<code>97</code>	<code>int</code>
<code>kLargePrime</code>	<code>16908799</code>	<code>int</code>
<code>kSmallPrime</code>	<code>127</code>	<code>int</code>
<code>last</code>	<code>@0x7fffffff120</code>	<code>str</code>

The bottom pane shows the Qt Thread Viewer, which displays the current thread state. The thread is stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	<code>nameHash</code>	<code>name_hash...</code>	<code>66</code>	1	<code>nameHash(...</code>	<code>/home/keit...</code>	<code>66</code>	<code>0x4c9cc3</code>			(all)
1	<code>Main</code>	<code>name_hash...</code>	<code>31</code>								
2	<code>Main</code>	<code>main.cpp</code>	<code>23</code>								
3	<code>startupMain</code>	<code>platform.cpp</code>	<code>2208</code>								
4	<code>main</code>	<code>name_hash...</code>	<code>27</code>								

The bottom status bar shows the Qt IDE interface, including the Qt logo and the text "Qt".

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

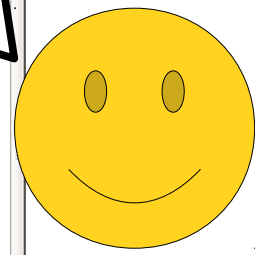
- main.cpp
- name\_hash.cpp

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the string
60  * name,
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71

```

Go click "Step Over" again to run this line of code.



Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	100 'd'	ch
first	@0x7fffffff100	str
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	@0x7fffffff120	str



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

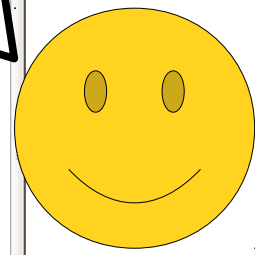
- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16999799;
55  static const int kSmallPrime = 97;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the string.
60  * name, first, last.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71

```

You should be here now. Notice that none of the values changed. That makes sense, since all we did was convert a lower-case 'd' to a lower-case 'd'.



Open Documents

- main.cpp
- name\_hash.cpp

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	100 'd'	ch
first	@0x7fffffff100	str
hashVal	97	int
kLargePrime	16998799	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name\_hash.cpp
  - Other files

Open Documents

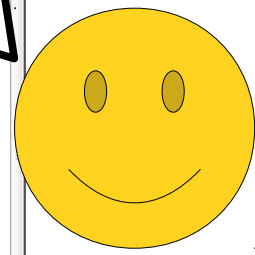
- main.cpp
- name\_hash.cpp

```

45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16999799
55  static const int kSmallPrime = 1000000007
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the range [first, last)
60  * name, first, last
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric value of
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71

```

Now, click "Step Over" one more time.



Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67								
1	Main	name_hash...	31	1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								





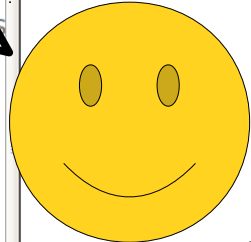




Qt IDE interface showing a C++ code editor with a breakpoint at line 62. A speech bubble explains that the breakpoint should be cleared for the program to run again in debug mode without stopping at that point. A yellow smiley face is drawn next to the bubble.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53
54
55
56
57
58
59
60
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

... it should clear the breakpoint. Now, if we were to run this program again in debug mode, it would not stop at this point, since nothing's telling it to!



Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62								
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Qt IDE interface showing a C++ project named 'name-hash'. The main editor window displays the source code for 'name\_hash.cpp'. The code defines a function 'nameHash' that calculates a hash value based on two input strings. The function uses two prime numbers, 'kLargePrime' (16908799) and 'kSmallPrime' (127), to calculate the hash. The code is as follows:

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric v
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71 }
```

The Qt IDE interface includes a sidebar with various tool icons (Welcome, Edit, Design, Debug, Projects, Analyze, Help), a 'Projects' pane showing the project structure, an 'Open Documents' pane, and a 'Debug' pane at the bottom. The 'Debug' pane shows a stack trace with the following entries:

Level	Function	File	Line
0	nameHash	name_hash...	62
1	main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name_hash...	27

A red box highlights the 'nameHash' function entry in the stack trace. A yellow smiley face is drawn next to the stack trace, and a speech bubble points to it with the text: "Now, take a look back at these buttons."

Qt IDE screenshot showing a C++ code editor with a function `nameHash`. The code includes comments and a loop over characters. A red box highlights the loop header on line 62. A yellow smiley face is next to the code. A speech bubble contains the text: "Hover your mouse over the one that's fifth from the left. When you hover over it, it should say 'step out.'" The right sidebar shows a variable watch window with a question mark next to `hashVal`.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric v
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	100 'd'	ch
first	fffffe100	str
hashVal	?	int
kLargePrime	99	int
kSmallPrime	127	int
last	@0x7fffffff120	str

Threads: #1 name-has

Level	File	Line
0	nameHash	name_hash... 62
1	Main	name_hash... 31
2	Main	main.cpp 23
3	startupMain	platform.cpp 2208
4	main	name_hash... 27

Qt IDE interface showing a C++ code editor with the following code:

```
45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less th
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the las
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric v
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

The Qt IDE interface includes a sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The 'Open Documents' pane shows 'main.cpp' and 'name\_hash.cpp'. The 'Threads' pane at the bottom shows a stack trace with a red box around the 'nameHash' function call at line 62. The 'Variables' pane on the right shows a table with a red box around the 'hashVal' variable, which has a question mark icon next to its value.

Name	Value	Type
__for_begin	@0x7fffffff030	str
__for_end	@0x7fffffff040	str
__for_range	<not accessible>	
ch	100 'd'	ch
first	ffffffe100	str
hashVal	?	int
kLargePrime	99	int
kSmallPrime	127	int
last	@0x7fffffff120	str

A yellow smiley face is drawn next to the 'hashVal' variable in the Variables pane. A speech bubble points to the 'nameHash' function call in the stack trace, containing the text: "If you click this button, it will keep running this function up until it completes and returns."

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name\_hash.cpp', which implements a name hashing function. The code includes comments explaining the hashing scheme and defines two prime constants: `kLargePrime = 16908799` and `kSmallPrime = 127`. The function `nameHash` iterates over characters in two strings, converting them to lowercase and updating a hash value using the formula `hashVal = (kSmallPrime * hashVal + ch) % kLargePrime`. A red box highlights the 'Run' button in the bottom toolbar, and a yellow smiley face is drawn next to it.

```
45  * F_p, where p is a large prime number, and evaluates that polynomial a
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers, a large prime and a
51  * prime. These numbers were chosen because their product is less th
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first name, then the las
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case. The numeric v
64  * lower-case letters are always less than 127.
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Threads: #1 name-has

Level	File	Line
0	nameHash	name_hash... 62
1	Main	name_hash... 31
2	Main	main.cpp 23
3	startupMain	platform.cpp 2208
4	main	name_hash... 27

1 Issues 2 Search Results 3 Application O

Now, go click that button. If you did everything right...



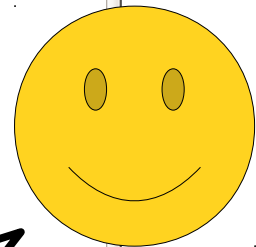
Projects

- name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
  - name\_hash.cpp
  - Other files

```

18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the act
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where p

```



... you should end up with something that looks like this!

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

Open Documents

- main.cpp
- name\_hash.cpp

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Type to locate (Ctrl...)

- 1 Issues
- 2 Search Results
- 3 Application Output
- 4 Compile Output
- 5 QML/JS Console
- 6 General Messages

Projects

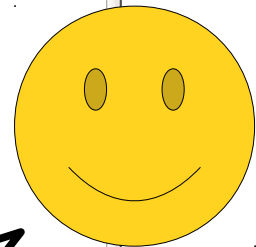
- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
      - src
        - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the act
38 * to talk mo
39 * the meanti
40 * of the inp
41 *
42 * For those
43 * treats eac
44 * It then us
45 * F n where p
```

Name	Value
first	@0x7fffffff0a0...
last	@0x7fffffff0c0...
hashValue	-590633613



Let's take a minute to get our bearings.  
Where exactly are we?

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

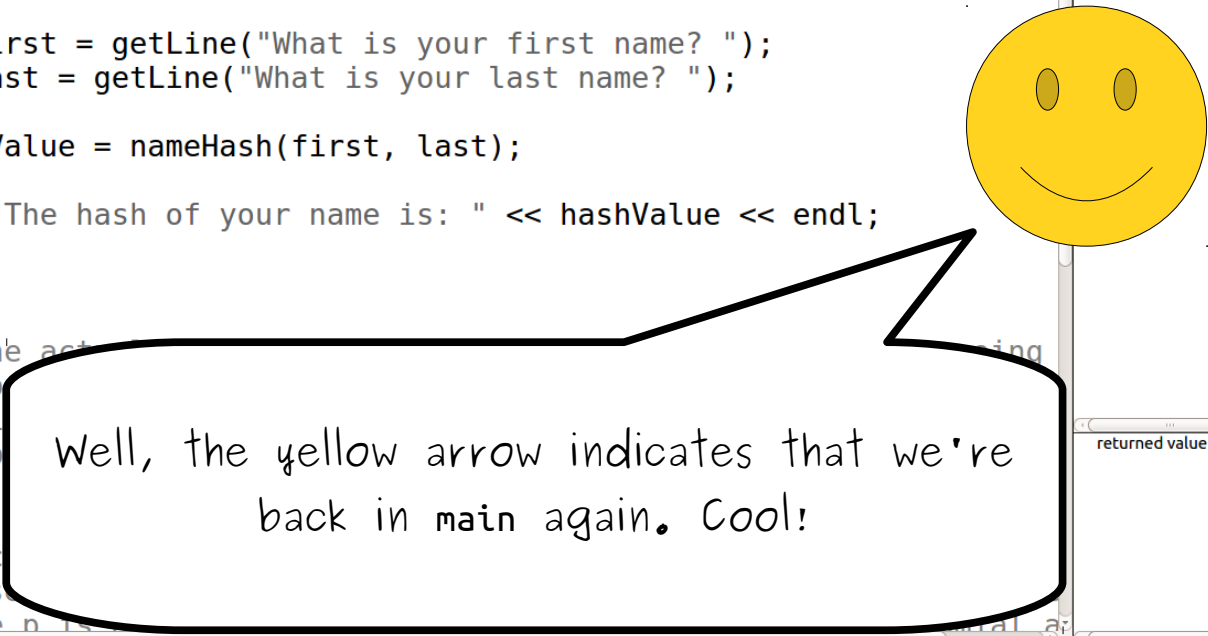
Qt IDE sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, Help, and a vertical toolbar with various development tools.

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the act
38  * to talk mo
39  * the meanti
40  * of the inp
41  *
42  * For those
43  * treats eac
44  * It then us
45  * F n where n
```

Qt IDE right sidebar showing a variable watch window with the following data:

Name	Value
first	@0x7fffffff0a0...
last	@0x7fffffff0c0...
hashValue	-590633613

Below the watch window, a status bar shows: returned value 1967457



Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Projects

- name-hash
  - name-hash.pro
    - Headers
    - Sources
      - lib/StanfordCPPLib
      - src
        - name\_hash.cpp
  - Other files

Open Documents

- main.cpp
- name\_hash.cpp

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34 }
35
36
37 /* This is the actual implementation of the nameHash function. It
38 * takes two strings as input and returns an integer representing the
39 * hash value. The hash value is calculated by treating each character
40 * of the input strings as a number and then combining them together.
41 * For those who are interested, the nameHash function treats each
42 * character as a number between 0 and 255. It then uses a simple
43 * algorithm to calculate the hash value. For example, if the input
44 * strings are "John" and "Doe", the hash value would be 1967457.
```



We can see that the nameHash function returned 1967457. Thanks, debugger!

Name	Value
first	@0x7fffffff0a0...
last	@0x7fffffff0c0...
hashValue	-590633613

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								



But if you look up over here in the values window, you can see that hashValue has some really weird-looking number stored in it. (You'll almost certainly see something different on your system.)

Qt IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name\_hash.cpp".

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p where p is a large prime number and evaluates that polynomial at
```

The "Values" window on the right shows the state of variables:

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

The "Debug Console" at the bottom shows the execution flow:

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

The status bar at the bottom indicates the application is stopped: "Stopped: 'function-finished'".

returned value 1967457



But it looks like we're setting hashValue equal to the number that was returned by the nameHash function. What's going on?

```
int hashValue = nameHash(first, last);
```

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and the function definition
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p where p is a large prime number and evaluates that polynomial at
```

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								



This is pretty cool, actually!

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name\_hash.cpp'. The code includes a function 'nameHash' and a 'main' function. A blue box highlights the line: `int hashValue = nameHash(first, last);`. The right sidebar shows a variable inspector with 'first' and 'last' variables. The bottom status bar shows the thread stack.

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for
23  * in main and
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38  * to talk more about what hash functions do later in the quarter. In
39  * the meantime, think of it as a function that scrambles up the characters
40  * of the input and produces a number.
41  *
42  * For those of you who are more mathematically inclined, this function
43  * treats each character in the input name as a number between 0 and 128
44  * It then uses them as coefficients in a polynomial over the finite field
45  * F_p where p is a large prime number and evaluates that polynomial at
```

Qt IDE interface showing a C++ project named 'name-hash'. The main editor displays the source code for 'name\_hash.cpp'. The code includes a function 'nameHash' and a 'main' function. A blue box highlights the line: `int hashValue = nameHash(first, last);`. The right sidebar shows a variable inspector with 'first' and 'last' variables. The bottom status bar shows the thread stack.

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27



What's happened is that we've just returned from nameHash with a value, but since we're going through the program one step at a time, we haven't actually assigned that value to hashValue yet!

```
int hashValue = nameHash(first, last);
```

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and this file
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last;
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p where p is a large prime number and evaluates that polynomial at
```

Name	Value
first	@0x7fffffff0a0
last	@0x7fffffff0c0
hashValue	-590633613

returned value 1967457

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								





Let's do a "step over" so that we can finish executing this line. Click "step over," and if you did everything right...

Qt IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name\_hash.cpp".

```
18 #include <string>
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashCode = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashCode << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38 * to talk more about what hash functions do later in the quarter. In
39 * the meantime, think of it as a function that scrambles up the characters
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p where p is a large prime number and evaluates that polynomial at
```

The Qt console shows the output: "The hash of your name is: 1967457".

The Debug Console shows the following stack trace:

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

The status bar at the bottom indicates the application is stopped: "Stopped: 'function-finished'".



... you should see the right value get stored (notice it's in red!) and we've moved to the next line.

```
File Edit Build Debug Analyze Tools Window Help
Projects
name-hash
  name-hash.pro
  Headers
  Sources
  lib/StanfordCPPLib
  src
  name_hash.cpp
  Other files
main.cpp
name_hash.cpp

20 using namespace std;
21
22 /* Prototype for the nameHash function. This is used
23  * in main and the nameHash function.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash code. We're going
38  * to talk more about what hash functions do later in the quarter. In
39  * the meantime, think of it as a function that scrambles up the characters
40  * of the input and produces a number.
41  *
42  * For those of you who are more mathematically inclined, this function
43  * treats each character in the input name as a number between 0 and 128
44  * It then uses them as coefficients in a polynomial over the finite field
45  * F_p, where p is a large prime number, and evaluates that polynomial at
46  * some smaller prime number q. (You aren't expected to know this for CS
47  * but we thought it might be fun!
48  */
```

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	1967457	int

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	33								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



Qt Creator IDE interface showing a C++ project named "name-hash". The main editor displays the source code for "name\_hash.cpp".

```
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23  * in main and then define it later in the program.
24  */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual function that computes the hash. We're going
38  * to talk more about what hash functions do later. In the meantime,
39  * think of it as a function that takes the character of the input
40  * and produces a number.
41  */
```

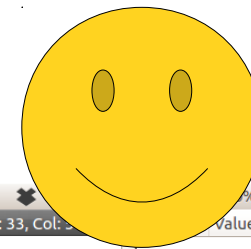
The Qt icon in the bottom-left toolbar is highlighted with a yellow box. A speech bubble points to it with the text: "To do this, click on this button. If you hover over it, it says 'Continue,' and that button means 'unpause the program and let it keep running from here.'" A yellow smiley face is also present next to the speech bubble.

On the right side, a variable viewer shows:

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	1967457	int

At the bottom, a stack of frames is visible:

Function	File	Line	Address	Condition	Ignore	Threads
main	main.cpp	27				
startupMain	platform.cpp	2208				
main	name_hash...	27				

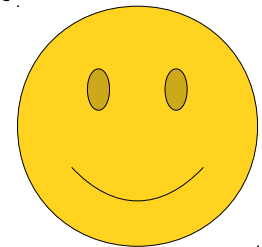


If you do, you should see something like this.  
(The program window might not automatically pop up. That's okay! Just open it manually.)  
Our program is now done running!

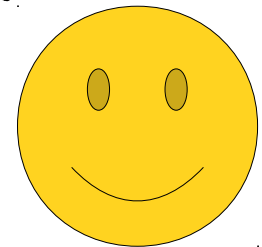
The screenshot shows the Qt Creator IDE interface. On the left is the Project Explorer showing a project named 'name-hash' with subfolders for 'name-hash.pro', 'Headers', 'Sources', 'lib/StanfordCPPLib', and 'src'. The 'src' folder contains 'name\_hash.cpp'. Below it is the 'Open Documents' list with 'main.cpp' and 'name\_hash.cpp'. The main editor window displays the source code for 'name\_hash.cpp' with line numbers 20 through 47. The code includes a `main` function that prompts for a first and last name and calculates a hash. A console window is overlaid on the code, showing the program's execution: 'What is your first name? Ada', 'What is your last name? Lovelace', and 'The hash of your name is: 1967457'. At the bottom, the 'Threads' panel shows 'Debugger finished.' and the 'Views' panel shows 'Application Output' selected.

```
20 int main() {
21     std::string first;
22     std::string last;
23     std::cout << "What is your first name? ";
24     std::getline(std::cin, first);
25     std::cout << "What is your last name? ";
26     std::getline(std::cin, last);
27     int hash = hash_name(first, last);
28     std::cout << "The hash of your name is: " << hash << endl;
29 }
30
31 /* This function calculates the hash of a name. It uses a polynomial
32 * to calculate the hash value. The polynomial is defined as:
33 * F_p, where p is a large prime number, and evaluates that polynomial at
34 * some smaller prime number q. (You aren't expected to know this for CS
35 * but we thought it might be fun!)
```

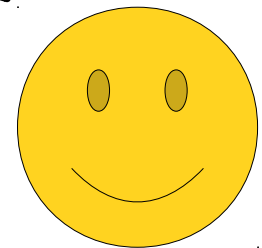
so there you have it! You've now gotten more familiar with the debugger!



You know how to set a breakpoint to pause the program at a particular point.

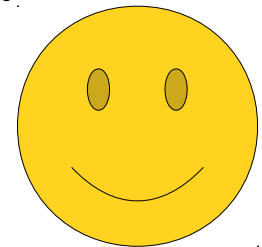


You know how to read the call stack and to see the values of local variables.

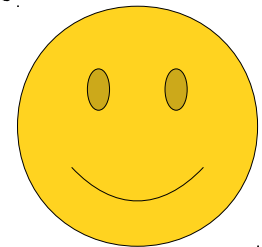




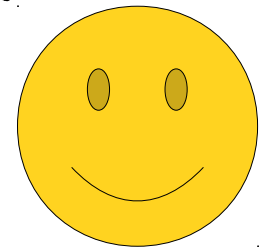
You know how to single-step the program and see what values change.



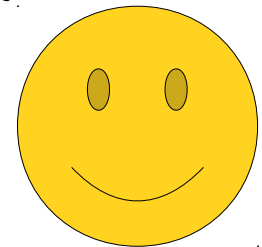
You know how to run a function to completion,  
and how to let the program keep on running.



As you write more and more complicated programs this quarter, you'll get a lot more familiar using the debugger and seeing how your programs work.



And, if you continue to build larger and larger pieces of software, you'll find that knowing how to use a debugger is a surprisingly valuable skill!



Hope this helps, and welcome to CS106B!

