C++ Library Reference Sheet

Lexicon

```
Lexicon lex; Lexicon english(filename);
lex.addWord(word);
bool present = lex.contains(word);
bool pref = lex.containsPrefix(p);
int numElems = lex.size();
bool empty = lex.isEmpty();
lex.clear();
```

Map

```
Map<K, V> map = {{k_1, v_1}, … {k_n, v_n}};
map[key] = value; // Autoinsert
bool present = map.containsKey(key);
int numKeys = map.size();
bool empty = map.isEmpty();
map.remove(key);
map.clear();
Vector<K> keys = map.keys();
```

Stack

```
 stack.push(elem);
 T val = stack.pop();
 T val = stack.top();
 int numElems = stack.size();
 bool empty = stack.isEmpty();
 stack.clear();
```

Queue

```
 queue.enqueue(elem);
 T val = queue.dequeue();
 T val = queue.peek();
 int numElems = queue.size();
 bool empty = queue.isEmpty();
 queue.clear();
```

Set

```
Set<T> set = {v_1, v_2, …, v_n};
set.add(elem);
set += elem;
bool present = set.contains(elem);
set.remove(x); set -= x; set -= set2;
Set<T> unionSet = s1 + s2;
Set<T> intersectSet = s1 * s2;
Set<T> difference = s1 - s2;
T elem = set.first();
int numElems = set.size();
bool empty = set.isEmpty();
set.clear();
```

Vector

```
Vector<T> vec = {v_1, v_2, …, v_n};
vec.add(elem);
vec += elem;
vec.insert(index, elem);
vec.remove(index);
vec.clear();
vec[index]; // Read/write
int numElems = vec.size();
bool empty = vec.isEmpty();
vec.subList(start, numElems);
```

TokenScanner

```
TokenScanner scanner(source);
while (scanner.hasMoreTokens()) {
   string token = scanner.nextToken();
   …
}
scanner.addWordCharacters(chars);
```

string

```
str[index]; // Read/write
str.substr(start);
str.substr(start, numChars);
str.find(c); // index or string::npos
str.find(c, startIndex);
str += ch;
str += otherStr;
str.erase(index, length);
```

ifstream

```
input.open(filename);
input >> val;
getline(input, line);
```

GWindow

```
GWindow window(width, height);
gw.drawLine(x0, y0, x1, y1);
pt = gw.drawPolarLine(x, y, r, theta);
```

GPoint

```
double x = pt.getX();
double y = pt.getY();
```

General Utility Functions

```
int getInteger(optional-prompt);
double getReal(optional-prompt);
string getLine(optional-prompt);
int randomInteger(lowInclusive,
                  highInclusive);
double randomReal(lowInclusive,
                  highExclusive);
error(message);
x = max(val1, val2); y = min(val1, val2);
stringToInteger(str); stringToReal(str);
integerToString(intVal);
realToString(realVal);
```