CS106X Instructor: Cynthia Lee

Autumn 2015 Practice Exam

PRACTICE FINAL EXAM: LIBRARY REFERENCE SHEET

This document has been updated from the midterm to include data structures we covered after the midterm.

Summary of Relevant Data Types

We tried to include the most relevant member functions for the exam, but not all member functions are listed. You are free to use ones not listed here that you know exist. **You do <u>not</u> need `#include`.**

```
class string {
 bool empty() const;
 int size() const;
 int find(char ch) const;
 int find(char ch, int start) const;
 string substr(int start) const;
 string substr(int start, int length) const;
 char& operator[](int index);
 const char& operator[](int index) const;
};

class Vector {
 bool isEmpty() const;
 int size() const;
 void add(const Type& elem); // operator+= used similarly
 void insert(int pos, const Type& elem);
 void remove(int pos);
 Type& operator[](int pos);
};

class Grid {
 int numRows() const;
 int numCols() const;
 bool inBounds(int row, int col) const;
 Type get(int row, int col) const; // cascade of operator[] also works
 void set(int row, int col, const Type& elem);
};
```

```
class Stack {

 bool isEmpty() const;
 void push(const Type& elem);
 Type pop();
};

class Queue {
 bool isEmpty() const;
 void enqueue(const Type& elem);
 Type dequeue();
};


class Map {
 bool isEmpty() const;
 int size() const;
 void put(const Key& key, const Value& value);
 bool containsKey(const Key& key) const;
 Value get(const Key& key) const;
 Value& operator[](const Key& key);
};
```

Example range-based for: `for (Key key : mymap){…}`

```
class Set {
 bool isEmpty() const;
 int size() const;
 void add(const Type& elem);
 bool contains(const Type& elem) const;
};
```

Operators:

```
set + value // Returns the union of set set1 and individual value value
set += value // Adds the individual value value to the set set
set1 += set2 // Adds all the elements from set2 to set1
```

Example range-based for: `for (Type elem : mymap){…}`

```
class Lexicon {
  int size() const;
  bool isEmpty() const;
  void clear();
  void add(std::string word);
  bool contains(std::string word) const;
  bool containsPrefix(std::string prefix) const;
};
```

Example range-based for: `for (string str : english){…}`

```
struct Edge {
    Vertex* start;
    Vertex* finish;
    double cost;
    bool visited;
};
struct Vertex  {
    std::string name;
    Set<Edge*> arcs;
    Set<Edge*>& edges;

    bool visited;
    Vertex* previous;
};

class BasicGraph : public Graph<Vertex, Edge> {
public:
BasicGraph();
    Vertex* addVertex(Vertex* v);
    const Set<Edge*>& getEdgeSet() const;
    const Set<Edge*>& getEdgeSet(Vertex* v) const;
    const Set<Edge*>& getEdgeSet(std::string v) const;
    Vertex* getVertex(std::string name) const;
    const Set<Vertex*>& getVertexSet() const;
    void removeEdge(std::string v1, std::string v2, bool directed = true);
    void removeEdge(Vertex* v1, Vertex* v2, bool directed = true);
    void removeEdge(Edge* e, bool directed = true);
    void removeVertex(std::string name);
    void removeVertex(Vertex* v);
}
```