CSCI 103 Introduction to Programming Fall 2014

Final Exam

For this exam, you are allowed to use a two-sided cheatsheet (8.5"x11") written in your own handwriting.

No calculators, computers, or textbooks are allowed.

All necessary `#include` directives, `using namespace std;` and the declaration/return of `main` are left out of many of the programs inside, but you should assume they are included.

Print your name, print your email address, and select your lecture section now.

Your Name: _____

Your USC e-mail: _____

Your Lecture Section: _____

29919	12:00PM MW	Mark Redekopp
30395	9:30AM TTh	David Pritchard
29920	11:00AM TTh	Massoud Ghyam
29922	12:30PM TTh	David Pritchard
29921	5:00PM TTh	David Pritchard

Problem	Value	Score
1	12	
2	16	
3	8	
4	10	
5	8	
6	8	
7	10	
8	6	
9	7	
10	12	
11	13	
Total	**110**	

1 True/False (12 points)

Circle each correct answer.

(a) If we run the lines

```
string A = "Hello";
string B = A;
B[0] = 'Y';
```

then at the end, the value of `A` is `"Yello"`.

true **false**

(b) If we run the lines

```
istringstream iss("text");
string s;
iss >> s;
bool b = iss.fail();
```

then at the end, the value of `b` is:

true **false**

(c) Adding an item to the front of a vector can be done in constant time.

true **false**

(d) Deleting the item at the end (back) of a vector can be done in constant time.

true **false**

(e) If we have an array of N `int`s that has been sorted, we can determine in $O(\log N)$ time whether it contains the number 103103103.

true **false**

(f) Statically allocated variables (local to functions) live in the stack, and dynamically allocated variables live in the queue.

true **false**

2 Data Types, Input/Output (16 points)

(a) For each of the two data types below, and each of the four properties listed, circle the correct answer (True or False).

	a dynamically allocated `int` array		a `vector<int>` object	
Memory is automatically deallocated	**true**	**false**	**true**	**false**
Length can be changed	**true**	**false**	**true**	**false**
For any k, can access kth integer inside of it in constant time	**true**	**false**	**true**	**false**

(b) For each of the two stream operations below, and each of the four properties listed, circle the correct answer (True or False).

	the `getline` function		the `>>` operator	
Skips all whitespace	**true**	**false**	**true**	**false**
Can write data directly to an `int`	**true**	**false**	**true**	**false**
Can cause stream to fail if there is no more data to read	**true**	**false**	**true**	**false**
Can be used with an `ofstream`	**true**	**false**	**true**	**false**

(c) Consider this code:

```
ostringstream oss;
for (int i = 5; i < 8; i++)
   oss << i;
istringstream iss(oss.str());
int x;
iss >> x;
```

What is the value of `x` after this code runs? _____

3 Terminology (8 points)

Find the best match of each term with the descriptions by writing a letter from 0 to 8 in each of the blanks. Each number should be used exactly once. One is filled in for you as an example.

accessor	_____
constructor	_____
data member	_____
destructor	_____
member function	_____
mutator	_____
private	_____
public	_____
recursive	**1**

0. a function inside of a class definition
1. a function that calls itself
2. a special kind of member function to change a private data member
3. a special kind of member function to read a private data member
4. a variable inside of a class definition
5. called when a variable of that class ceases existence (i.e. deallocated)
6. called when a variable of that class comes into existence (i.e. allocated)
7. the parts of a class definition that any code can access
8. the parts of a class definition that only code from that class can access

4 Performance (10 points)

(a) Consider the following code that is trying to determine if the elements of a vector are distinct.

```
1. // assume v is a vector<int> of length n
2. bool all_distinct = true;
3. for (int i = 0; i < n; i++)
4.    for (int j = 0; j <= i; j++)
5.       if (v[i] == v[j])
6.          all_distinct = false; // found two equal items
```

What order of growth best represents the running time of this code? Circle the best answer.

$O(\log n)$ $O(n)$ $O(n \log n)$ $O(n^2)$ $O(n^3)$ $O(2^n)$

(b) The code fragment shown actually has a bug: it gives the wrong answer sometimes. Give an example of a length-3 integer vector upon which this code gives the wrong result:

{_____, _____, _____}

(c) By adding, deleting or changing at most one character, fix the bug. Indicate what line number in the source code must change.

Edit line number: _____. Make this change: _____

(d) Elaine works on this problem for a while and comes up with another program to do the same thing, but using a faster approach. She measures the running time of her new program on different inputs and obtains the following table:

n	running time
2 million	25 seconds
5 million	1 minute
10 million	2 minutes

Assume that the running time is approximately of the form $a \times N^b$ where b is an integer. What is b? _____

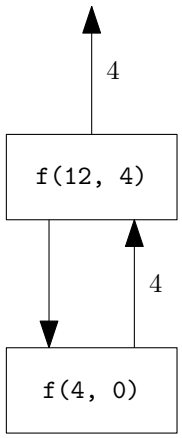
(e) Estimate the running time of her program when n is 80 million. You can leave your answer in either minutes or seconds.

Estimated running time: _____

5 Recursion (8 points)

Suppose we define the following function:

```
int f(int x, int y) {
  if (y == 0) return x;
  else return f(y, x%y);
}
```



(a) Draw the recursive call tree for `f(36, 15)` in the space below. A sample recursive call tree for a different set of inputs is shown on the right. Including return values on the diagram is optional.

(b) What is the return value of `f(36, 15)`? _____

(c) What is the return value of `f(200, 60)`? _____

6 Recursion (8 points)

Consider the following function.

```
string recur(string s) {
   int len = s.length();
   if (len <= 1)
      return s;
   else {
      string left = recur(s.substr(0, len/2)); // recur on left half
      string right = recur(s.substr(len/2, len/2)); // recur on right half
      cout << right + left << endl; // concatenate right, THEN left
      return right + left; // return concatenated results
   }
}
```

(a) Draw the recursive call tree that occurs when we call `recur("snow")`. Including local variables or return values on the diagram is optional.

(b) What three lines of output does `recur("snow")` print?

_____

_____

_____

7 Data Types (10 points)

Consider the following code:

```
 1: deque<int> d;
 2: int x;
 3: while (cin >> x) { // loop through the numbers on cin
 4:    if (x > 0)
 5:       d.push_front(x);
 6:    else {
 7:       cout << d.front() << " ";
 8:       d.pop_front();
 9:    }
10: }
```

(a) What output does this program print, if the standard input is `5 10 -1 15 20 -1`

(b) What elements, from front to back, remain in the deque at the end of the program?

(c) What output does this program print, if the standard input is `5 10 -1 15 20 -1` **and** on line 5 we replace `push_front` with `push_back`?

(d) What elements, from front to back, remain in the deque at the end of the modified program?

---

8 Dynamic Memory (6 points)

Consider the following incomplete C++ program:

```
const int N = 100;

struct Thing {
   Thing* x;
};

Thing* allocate_things() {
   Thing* t = new Thing[N];
   for (int i=0; i<N; i++) {
      t[i].x = new Thing;
      t[i].x->x = NULL;
   }
   return t;
}

void deallocate_things(Thing* addr);

int main() {
   Thing* addr = allocate_things();
   deallocate_things(addr);
}
```

Complete the function `deallocate_things(Thing* addr)` so that the program has no memory leak.

```
void deallocate_things(Thing* addr) {
// your code here









}
```

9 Object Oriented Programming (7 points)

Suppose we are creating a `Fraction` class to represent a rational number like 4/5 or 11/3 or −6/1. It should support the following operations:

```
// operation 1. a constructor with given numerator and denominator
Fraction f(4, 5);
Fraction f2(2, 3);
// operation 2. a function to tell if a fraction is positive
bool b = f.is_positive(); // returns true
// operation 3. a function to compute the sum of two fractions
Fraction f3 = f.plus(f2);
```

Write a class declaration (the kind you would find in a header file) for this class. You do not have to actually implement the functions (there's no need to write the .cpp file). However, you should include any necessary data members (don't add unnecessary ones). **Represent the data exactly using `int` variables rather than `long`s or `double`s.**

```
#ifndef FRACTION_H
#define FRACTION_H
class Fraction {









};
#endif
```

10 Object Oriented Programming (12 points)

In this exercise you will write a .cpp file to implement a `Point` class that represents a point in 2-dimensional Euclidean space, supporting the operations listed below.

- Write out your solution on the **next** page.
- You may carefully rip *this page* (not the *next page*) out of the exam to use for reference while you complete your solution. Be very careful not to rip out any other pages.

Here are the operations:

```
// operation 1. a constructor with given x and y coordinates
Point p(1.5, 2.0);

// operation 2. get a string representation
cout << p.as_string() << endl; // "(1.5, 2)" in this case

// operation 3. check if two points have the same location
Point q(2.0, 1.5);
bool b = p.equal_to(q); // returns false

// operation 4. transform the point by swapping its coordinates
p.reflect();
cout << p.as_string() << endl; // now gives "(2, 1.5)"
bool b2 = p.equal_to(q);       // now true
```

You should assume `point.h` is defined as follows:

```
// this is point.h
#ifndef POINT_H
#define POINT_H
#include <string>
using namespace std;

class Point {
   public:
      Point(double xpos, double ypos); // operation 1
      string as_string();              // operation 2
      bool equal_to(Point other);      // operation 3
      void reflect();                  // operation 4
   private:
      double x;
      double y;
};
#endif
```

Write your solution to problem number 10 here.

```
// this is point.cpp
#include <string>
#include <sstream>
#include "point.h"

// operation 1









// operation 2. use default numeric formatting; setw/manipulators not needed









// operation 3









// operation 4
```

11 Linked Lists, Recursion (13 points)

Suppose we have a singly-linked list with a head pointer, using the following struct and class.

```
struct Node {
   Node* next;
   int val;
};
```

```
class List {
 private:
   Node* head;
   void helper(int target, Node* curr);
 public:
   void deleteTarget(int target);
};
```

Fill in the definition of `deleteTarget` and its helper function below, so that `deleteTarget` deletes a target value from a linked list. For example if the list contains 1, 5, 6, 7 and we delete 6, then the list should contain 1, 5, 7. Your code should deallocate the removed node.

- You must use recursion and you **must not use any loop**.
- Clarification: If the value doesn't appear, the linked list should be unaffected. If the value appears more than once, *only the first* occurrence should be deleted.
- If you need more space use the next page or the back of the previous. *Mention this below!*

```
void List::deleteTarget(int target) { // delete target from list, if it exists
   if (                                          ) {   // blank 1
      return;
   }
   else if (head->val == target) {
                                                       // blank 2




   }
   else helper(target, head);
}

void List::helper(int target, Node* curr) {
   if (                                          ) {   // blank 3
      return;
   }
   else if (                                     ) {   // blank 4
                                                       // blank 5



   }
   else helper(                                     ); // blank 6
}
```

You can carefully tear this page out and use it for scratch work. If you do anything on it you want graded, clearly indicate this on the appropriate page and put this sheet back in the exam at the end, writing your name on it and clearly labeling your work.