CSCI 103 Introduction to Programming Spring 2015

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.

Some of the programs inside omit their `#include` and `using` statements and the declaration/return from `main`. They are only omitted for brevity. Please assume that 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: _____

29991	2PM MW	David Pritchard
29902	11AM TTh	David Pritchard
29906	2PM TTh	David Pritchard

Problem	Value	Score
1	12	
2	9	
3	11	
4	8	
5	8	
6	12	
7	10	
Total	**70**	

1 Syntax and Semantics (12 Points)

Circle **True** or **False** for each option.

(a) What can the `>>` operator be used for?

- checking if one number is a lot bigger than another, e.g. `if (x >> limit)`

  True **False**

- reading a `double` out of an `istringstream`

  True **False**

- shifting an integer to the right by a certain number of binary positions

  True **False**

- declaring a 2D vector, e.g. `vector<vector<int>> grid`

  True **False**

(b) Consider the following three kinds of function prototypes:

```
void func(int x);  // normal parameter
void func(int* x); // pointer parameter
void func(int& x); // reference parameter
```

For what reason(s) would we define the function using a pointer parameter or a reference parameter instead of a normal parameter?

	pointer	reference
function can change an `int` variable's value	**True**/**False**	**True**/**False**
allows us to pass an array to the function	**True**/**False**	**True**/**False**
is valid syntax to call it on `int` variable (e.g., `func(myint)`)	**True**/**False**	**True**/**False**
avoids making copy of `int` when we call the function	**True**/**False**	**True**/**False**

2 OOP and Dynamic Memory (9 Points)

Circle **True** or **False** for each option.

(a) Member functions always have to be `public`.

True **False**

(b) If `s1` and `s2` are `structs`, `s1 == s2` checks if all their data members are identical.

True **False**

(c) It is possible for a class to have several different constructors.

True **False**

(d) It is possible for a class to have several different destructors.

True **False**

(e) Consider the following code snippet:

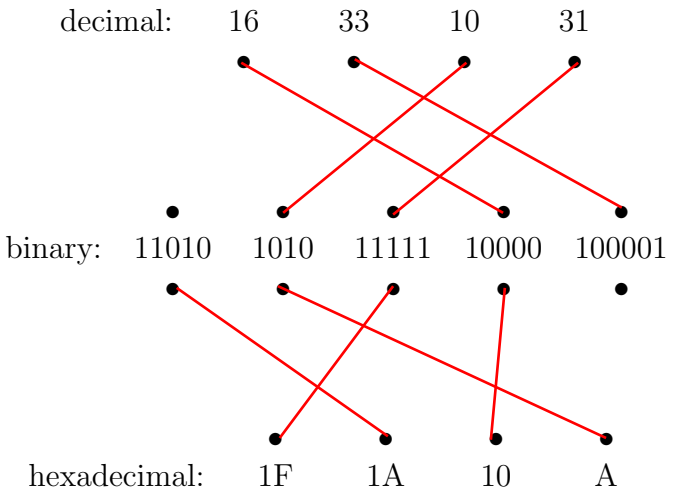
```
int a[2];
int* x = new int[3];
int* y = new int[5];
int* t = y;
y = x;
x = a;
delete t;
delete y;
delete x;
```

This code compiles. However, what bug(s) does it have?

	code has this bug
can't `delete t`, since it was not initialized using `new`	**True**/**False**
has memory leak	**True**/**False**
mixes arrays of different lengths	**True**/**False**
tries to deallocate a statically-allocated array	**True**/**False**
wrong deallocation syntax	**True**/**False**

3 Number Systems and Bitwise Operators (11 Points)

(a) Match each decimal number to the equivalent binary number, and match each hexadecimal number to the equivalent binary number. Indicate the matched values by connecting the dots; you should draw 8 lines in total.



(b) Compute the results of the following bitwise operations. *All numbers below are binary, and you should leave your answers in binary.*

(i) `1101` $\wedge$ `1011`

110

(ii) `101010 & 111000`

101000

(iii) `11100 << 11`

11100000

4 Orders of Growth (8 Points)

Circle the order of growth for each description below.

Every answer should be used exactly once.

(a) The amount of memory used by a `Picture` with height N and width N.

1 (constant) $\log N$ N $N \log N$ **N^2** N^3 2^N 3^N

(b) The time it takes to draw a Sierpinski triangle of order N.

1 (constant) $\log N$ N $N \log N$ N^2 N^3 2^N **3^N**

(c) The time it takes to find the middle element of a vector of length N.

1 (constant) $\log N$ N $N \log N$ N^2 N^3 2^N 3^N

(d) The time it takes to find the middle element of a linked list of length N.

1 (constant) $\log N$ **N** $N \log N$ N^2 N^3 2^N 3^N

(e) The number of different N-bit binary integers.

1 (constant) $\log N$ N $N \log N$ N^2 N^3 **2^N** 3^N

(f) The time it takes for the following code to execute:

```
for (int i=0; i<N; i++) for (int j=1; j<N; j*=2) cout << i+j << endl;
```

1 (constant) $\log N$ N **$N \log N$** N^2 N^3 2^N 3^N

(g) The running time of a program that takes 1 hour when N is 1000, 1.125 days when N is 3000, and 9 days when N is 6000.

1 (constant) $\log N$ N $N \log N$ N^2 **N^3** 2^N 3^N

(h) The running time of binary search on an array of N elements.

1 (constant) **$\log N$** N $N \log N$ N^2 N^3 2^N 3^N

5 Recursion (8 Points)

Consider the following recursive function:

```
int f(int m, int n) {
   cout << "C " << m << " " << n << endl; // C: call
   int result = n;
   if (m > 0)
      result = f(m - 1, m + n)
   cout << "R " << result << endl; // R: return
   return result;
}
```

(a) In the box below, list all lines of output that are printed if we call `f(3, 3)`.

Lines of output:

```
C 3 3
C 2 6
C 1 8
C 0 9
R 9
R 9
R 9
R 9
```

(b) What is the return value of `f(3, 3)`? 9

(c) If `n` is any positive integer, what is the return value of `f(n, 0)`, in terms of `n`? You do not need to simplify your answer; it is okay to write an expression involving dots (···).

1 + 2 + ... + n

6 OOP Debugging (12 Points)

Three files are listed below. On the next page, you will help debug them.

You may carefully detach this page from the exam if you want to look at the pages side-by-side. Please don't rip out the staple!

```
   /*** circle.h: ***/
 1 class Circle {
 2  private:
 3
 4   // x-center, y-center, radius of a circle
 5   double x, y, r;
 6
 7   // construct a circle with the given center coordinates and radius
 8   Circle(double xcenter, double ycenter, double radius);
 9
10   // check if the point (px, py) is contained inside of this circle
11   bool contains(double px, double py);
12
13 };
```

```
   /*** circle.cpp: ***/
 1 #include "circle.h"
 2 Circle::Circle(double xcenter, double ycenter, double radius) {
 3    xcenter = x; ycenter = y; radius = r;
 4 }
 5
 6 bool contains(double px, double py) {
 7    return (px-x)*(px-x) + (py-y)*(py-y) <= r*r; // (FORMULA IS CORRECT)
 8 }
```

```
   /*** test_circle.cpp: ***/
 1 #include "circle.h"
 2 #include <iostream>
 3 #include <iomanip>
 4 using namespace std;
 5 int main() {
 6    Circle mycircle(1, 2, 100); // center (1, 2), radius 100
 7    cout << boolalpha << mycircle.contains(5, 10) << endl;
 8 }
```

OOP Debugging, Continued

Answer the following questions about the code on the previous page. Explanations should be clear and brief (at most ten words if possible). *Writing the fixed code is not required.*

Hint: line 7 of `circle.cpp`, which says `FORMULA IS CORRECT`, has no bugs (it is not a trick).

(a) When we compile `circle.cpp`, we get the error `"use of undeclared identifier 'x'"` reported inside of the `contains` function in `circle.cpp`.

What file needs to be edited to fix this bug? *Circle one.*

`circle.h` **`circle.cpp`**

What line number in that file would you change? 6

What needs to be changed to fix it?

add Circle:: prefix before contains

(b) After fixing that issue, `circle.cpp` compiles. Then we try to compile `test_circle.cpp` but get the error `"calling a private constructor of class 'Circle'"`

What file needs to be edited to fix this bug? *Circle one.*

`circle.h` `circle.cpp` `test_circle.cpp`

What line number in that file would you change? 6

What needs to be changed to fix it?

add public:

(c) After fixing that issue, `test_circle.cpp` compiles. However, when we run it, it prints `false`, which is wrong since `mycircle` should be large enough to contain `(5, 10)`.

What file needs to be edited to fix this bug? *Circle one.*

`circle.h` **`circle.cpp`** `test_circle.cpp`

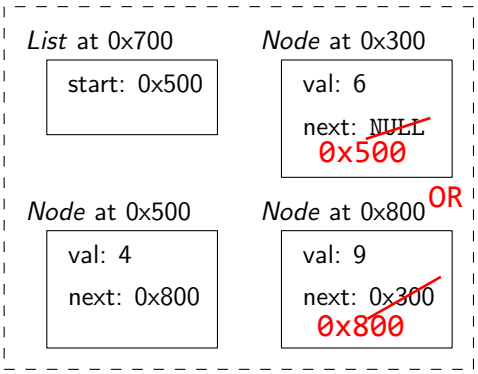
What line number in that file would you change? 3

What needs to be changed to fix it?

switch order of constructor parameters and data members
(change to x = xcenter; etc)

7 Linked Structures (10 Points)

Consider the following linked list code:




```
struct Node {
   int val;
   Node* next;
   Node(int v, Node* n) {val = v; next = n;}
};
class List {
 private: Node* start;
 public:
   void add() {
      for (int i=1; i<=2; i++) start = new Node(i, start);
   }
   bool mystery() {
      Node* a = start;  Node* b = start;
      while (b != NULL && b->next != NULL) {
         b = b->next;
         b = b->next;
         a = a->next;
         if (a==b) return true;
      }
      return false;
   }
// ... rest of List class definition not shown
```

(a) *Complete the following sentences about the* ***add()*** *function.*
The `add()` function adds two nodes to the **start** / **end** (*circle one*) of the list.
The new node with value 2 is **before** / **after** (*circle one*) the new node with value 1.

(b) We call `mystery()` on the linked list pictured at top right. Complete the trace of its execution below. Write down each change in the value of `a` or `b`, and the return value.

Initially, `a` is `0x500` and `b` is `0x500`.
Then, **a** / **b** (*circle one*) changes to 0x800.
Then, **a** / **b** (*circle one*) changes to 0x300.
Then, **a** / **b** (*circle one*) changes to 0x800.
The return value is **True** / **False** (*circle one*).

(c) Change *one* pointer in the diagram at top right so that if we ran `mystery()` on the modified structure, it would return the opposite value from part (b). *Cross out the value of the pointer you want to change; write the modified value clearly underneath it.*
Any change that creates a loop works.
For your reference, 2 of the 6 possible solutions are shown at top right.

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, writing your name on it and clearly labeling your work.