CSCI 103	Introduction to Programming	Fall 2014

Midterm Exam

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

No calculators, computers, or textbooks are allowed.

The lines `#include <iostream>`, `using namespace std;` and the declaration/return of `main` are left out of most 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**
0*	1	
1	10	
2	10	
3	9	
4	7	
5	9	
6	9	
7	8	
8	10	
9	12	
10	15	
Total	**100**	

* Problem 0: fill out this cover page.

1 Expressions (10 points)

What is the type and value of each of the following expressions? If it would not compile, write X in both boxes.

Expression	Type	Value
`1 * 1 + 2 * 2`		
`(double) (3 / 4)`		
`3 / 3.0`		
`(4 > 5) || (5 < 4)`		
`false == false == false`		

2 True/False (10 points)

Circle each correct answer.

(a) Anything that can be done with a `while` loop could be done with a `for` loop instead.

true **false**

(b) A function body may contain more than one `return` statement.

true **false**

(c) If `a` is an array, then `a[1]` is the same as `(*a)+1`.

true **false**

(d) If `ptr` is an `int*`, then `ptr+5` is the address 5 bytes after `ptr`.

true **false**

(e) The expression `new int` returns an `int` pointer rather than an `int`.

true **false**

3 Short Answer / Multiple Choice (9 points)

(a) When using command line arguments, what is the data type of `argv[1]`?

____

(b) When declaring/defining a function, if some input parameter `x` is a multidimensional array, which dimension sizes of `x` do you **have to** list?
Circle only one answer.

(i) all dimension sizes
(ii) only the first dimension size
(iii) only the last dimension size
(iv) all dimension sizes except the first dimension
(v) all dimension sizes except the last dimension

(c) Billy Bruin wrote this code, which is supposed to create the string `"****"`:

```
char str[50];
for (int i = 0; i < 4; i++) str[i] = '*';
```

What problem(s) does it have that can cause a bug?
Circle all that apply.

(i) Billy should fill all 50 characters of the array.
(ii) Billy should set the elements of the array to `"*"` instead of `'*'`.
(iii) Billy should add a null character immediately after the last `*`.
(iv) Billy should use the `new` operator to create the array.

(d) Remember that when you pass an array to a function, the argument is just the name of the array (e.g., `printHistogram(testCounts)` or `showRGBBMP(image)`). Describe, in at most 10 words, what information is really being passed to the function when we call it on an array name in this way.

____

____

4 Code Analysis (7 points)

Assume that `first` and `second` are two variables of `int` type, with unknown values. We run the following code:

```
1:  if (first < second) {
2:     cout << 'A';
3:     if (first > 20)
4:        cout << 'B';
5:  }
6:  else if (first > second) {
7:     cout << 'C';
8:     if (first > 32)
9:        cout << 'D';
10:  }
11: else {
12:    cout << 'E';
13: }
14: cout << endl;
```

(a) Given that the values of `first` and `second` are unknown, which possible lines of output could have been produced when this code is executed? *Circle all possible options that could have been printed.*

(i) `A` (ii) `B` (iii) `C` (iv) `D` (v) `E`

(vi) `AB` (vii) `AC` (viii) `CD` (ix) `BD` (x) `ACE`

(b) Suppose we change the `else if` on line 6 to `if`. What two options become possible that were not possible before? (They are not in the above list.)

5 Code Tracing (9 points)

Consider the following code fragment.

```
1:   int inputs[7] = {0, 3, 2, 2, 1, 2, 0};
2:   int freqs[4] = {0, 0, 0, 0};
3:   for (int i=0; i<7; i++) {
4:      freqs[inputs[i]] += 1;
5:   }
6:
7:   for (int i=0; i<4; i++) {
8:      for (int j=0; j<freqs[i]; j++) {
9:         cout << i << " ";
10:     }
11:  }
12:  cout << endl;
```

(a) When we reach line 6, what are the values in `freqs`?

`freqs[0]`: ________ `freqs[1]`: ________ `freqs[2]`: ________ `freqs[3]`: ________

(b) What line of output does the program produce?

____

(c) Describe, in ten words or less, what this program does. (I.e., describe the relationship between `inputs` and the printed output, in general.)

____

____

6 Debugging (9 points)

Here is a buggy program. It is supposed to take as input a number `n` followed by `n` more integers, and then print out their average. For example if the input is

```
3
1 2 4
```

then the output should be `2.33333` (which is $\frac{1+2+4}{3}$).

```
1:     int n;
2:     cin >> n;
3:     double total = 0.0;
4:     int count = 0;
5:     for (int i=0; i<n; n++) {
6:        double val;
7:        cin >> val;
8:        total == total + val;
9:     }
10:    cout << "The average was: " << (val/n) << endl;
```

The program has 3 bugs. For each bugs described below, indicate the line number it occurs on, and describe how to fix it in at most 10 words. (Each fix is a simple change on a single line, you should not define new variables or make complex changes.)

(a) When we try to compile the program it gives us a "scope" error.

Bug is on line number: ________

To fix this error: ______________________________

____

(b) When we fix that error, it compiles. When we run the program on the test input above, it loops infinitely! What bug is causing this?

Bug is on line number: ________

To fix this error: ______________________________

____

(c) When we fix that error, we run the program on the test input above. It prints out `0` instead of the correct output. What bug is causing this?

Bug is on line number: ________

To fix this error: ______________________________

____

7 Passing Semantics (8 points)

Consider the following program.

```
1:  void two(int *s, int t) {
2:     t += *s;
3:     *s += t + 1;
4:     t += 4;
5:  }
6:
7:  int main() {
8:     int u = 4, v = 3;
9:     two(&u, v);
10:    return 0;
11: }
```

(a) What are the values of `*s` and `t` at the end of the call to `two` (i.e., at line 5)?

`*s`: __________

`t`: __________

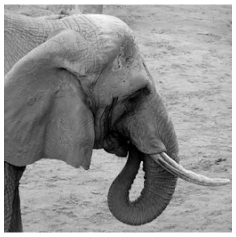
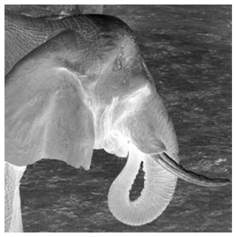
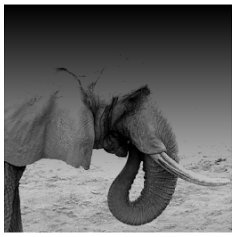
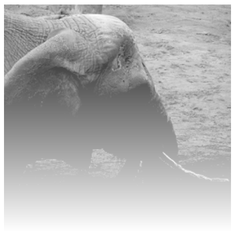
(b) What are the values of `u` and `v` when the `main` function returns (i.e., at line 10)?

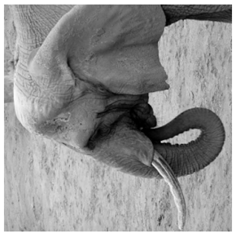
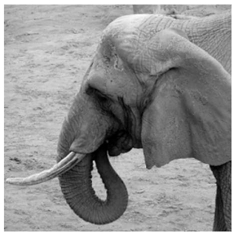
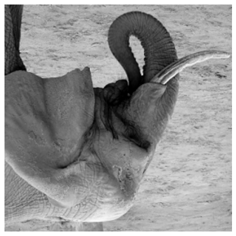
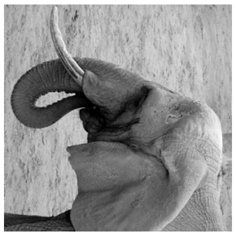
`u`: __________

`v`: __________

8 Graphics (10 points)

In this grid, **A** is the picture `elephant.bmp` from class, and 7 other images are shown.

A: 
B: 
C: 
D: 

E: 
F: 
G: 
H: 

Below, there is a program missing a line, and 6 possible lines to replace the missing one. For each possible replacement, indicate which picture would be shown if we ran the program using that line as the missing line. *Some pictures may occur more than once or not at all.* Remember that `SIZE` is defined to be equal to `256`.

```
unsigned char src[SIZE][SIZE];
unsigned char dest[SIZE][SIZE];
readGSBMP("elephant.bmp", src);
for (int i=0; i<SIZE; i++) {
   for (int j=0; j<SIZE; j++) {
      // MISSING LINE
   }
}
showGSBMP(dest);
```

Missing Line	Letter of Resulting Image
`dest[i][j] = src[i][j];`	
`dest[i][j] = min(i, src[i][j]);`	
`dest[255-i][j] = src[255-i][j];`	
`dest[i][j] = 255-src[i][j];`	
`dest[i][j] = src[i][255-j];`	
`dest[i][j] = src[j][i];`	

9 Composition (12 points)

Write a function `void reverse(char* dest, char* src)` that puts the reverse of the C string `src` into the C string `dest`. For example

```
char word1[] = "reward";
char word2[80];
reverse(word2, word1);
cout << word2;
```

should print out `drawer`.
You may assume that we have included `<cstring>` so that the utility function `strlen` is available. You may also assume that `src` points to a valid null-terminated character array and `dest` is large enough to hold the desired string. If you run out of space, write your solution on the back of the previous page.

```
void reverse(char* dest, char* src) {









}
```

10 Composition (15 points)

Define a function `extremes` to find the minimum and maximum values in an array of `int`s. Specifically, your function should take 4 arguments: the array itself, the length of the array, and pointers where it should write the min and max. For example, if we test your function with

```
const int len = 6;
int data[len] = {6, 3, 9, 5, 1, 8};
int min, max;
extremes(data, len, &min, &max);
cout << min << " " << max << endl;
```

then it should print out `1 9`. Assume the second argument (the array length) is always greater than or equal to 1. *Your function must not alter the array.* If you run out of space, write your solution on the back of the previous page **and clearly indicate below that you have done so**.

PRINT your name here: ______________________

This page intentionally left blank for scratch paper. Return it with your test.