CSCI 2100: Data Structures, Fall 2020

First Midterm Solution — Oct. 14, 2020

Name:	Email Address: @slu.edu

---

1. This exam is closed book and no calculating devices of any type will be allowed. You are allowed, however, to prepare in advance the front and back of the hand-out info page with whatever notes you wish to place on it, and you may use this page during the exam. When the exam is over, submit this sheet with the rest of your exam.
2. Print your full name and your email address in the boxes above.
3. Print your name at the top of every page.
4. Please write clearly and legibly. If I can't read your answer, I can't give you credit.
5. Remember, these are NOT necessarily in order of difficulty. Please read all the problems first, and don't allow yourself to get stuck on a single problem.

#	1	2	3	4	5	6	Total
Max	15	20	20	20	20	15	100
Score							

1. (15 points) Fill in the diagram below to represent the underlying memory configuration that is present after the following commands are executed (Note: identifiers should be represented by $\{a, b, c, d, e\}$ not containing pointer($*$) or reference(&) symbols). Be careful to use a lower case (variable name) and an upper case (character value). If the identifier or value slot is empty, then write down "empty".

```
char a('X');
char *b(&a);
char *c = new char('Y');
char d = *c;
```

Identifiers	Value	Address
a	X	241
b	241	242
c	244	243
	Y	244
d	Y	245
		246
		247
		248

2. (20 points) Below is a simple class written in C++, and on the next page is a main function which uses the class. Predict the output this code gives when run, assuming that the appropriate libraries (such as iostream) have been imported at the beginning.

```
class MyThing {

private:
  int a;
  int b;

public:

  MyThing() : a(0),b(0) {
    cout << "Thing constructed!" << endl;
  }

  int change(int c, int& d) {
     a = c;
     b = d;
     d = d+2;
     c = c+2;
  }

  int reallyChange(int& e, int& f) {
     e = e + a;
     f = f + b;
     change(e,f);
  }

  void printMe() {
     cout << "a is " << a << endl;
     cout << "b is " << b << endl;
  }
};
```

```
int main() {
   MyThing stuff;
   int x = 2;
   int y = 3;
   stuff.printMe();
   stuff.change(x,y);
   stuff.printMe();
   cout << "x is " << x << endl;
   cout << "y is " << y << endl;
   stuff.reallyChange(x,y);
   stuff.printMe();
   cout << "x is " << x << endl;
   cout << "y is " << y << endl;
}
```

The output will be as below:

```
Thing constructed!
a is 0
b is 0
a is 2
b is 3
x is 2
y is 5
a is 4
b is 8
x is 4
y is 10
```

3. (20 points) Suppose we have an initially empty stack S and an initially empty queue Q. Fill in the following table for the following sequence of commands. S should be (bottom –¿ top) and Q should be (front ¡– rear). Use the comma to have multiple elements, but do not use any space between elements (do not have any space after the comma). Write down "empty" if there is no element.

Operation	S bottom → top	Q front ← rear
Q.push(11)	( )	(11)
S.push(6)	(6)	(11)
S.push(Q.front())	(6, 11)	(11)
S.push(7)	(6, 11, 7)	(11)
Q.push(S.top())	(6, 11, 7)	(11, 7)
Q.pop()	(6, 11, 7)	(7)
S.pop()	(6, 11)	(7)
Q.pop()	(6, 11)	( )
Q.push(S.top())	(6, 11)	(11)
S.push(Q.front())	(6, 11, 11)	(11)

4. (20 points) Implement and complement the recursive code to compute the product of two positive integers, x and y, using only addition and/or subtraction. The function will take as its arguments two integers to multiply together (x x y) and will return the product.

   Hint: consider the following:

   6 x 1 = 6
   6 x 2 = 6 + (6 x 1)
   6 x 3 = 6 + (6 x 2) = 6 + [6 + (6 x 1)] = 6 + 6 + 6

```
#include<iostream>
using namespace std;

int product(int x, int y)
{
    // if x=0 or y=0, return 0
    if (x == 0 || y == 0)
        return 0;
    else if (y == 1) // if y=1, return x
        return x;
    else  // iteratively calculate y times sum of x
        return (x + product(x, y - 1));
}

int main()
{
    int x = 40, y = 10;
    cout << "product(" << x << ", " << y << ")=";
    cout << product(x, y) << endl;
    return 0;
}
```

5. (10 points) Give an analysis of the running time (Big-Oh notation) for each of the following program fragments

   (a)
   ```
   sum = 0;
   for(i=0; i<2n; i++)
       for(j=0; j<i; j++)
           sum++;
   ```

   Sol: $O(N^2)$

   (b)
   ```
   sum = 0;
   for(i=0; i<n; i++)
       for(j=0; j<i*i; j++)
           for(k=0; k<j; k++)
              sum++;
   ```

   Sol: j can be as large as i^2, which could be as large as N^2. k can be as large as j, which is N^2. The running time is thus proportion to $N \cdot N^2 \cdot N^2$, which is $O(N^5)$.

6. (15 points) Suppose that two entries of an array A are equal to each other. After running the insertion-sort algorithm of the below code, will they appear in the same relative order in the final sorted order or in reverse order?

   Explain your answer.

```
Algorithm InsertionSort(A):

   Input: An array A of n comparable elements
   Output: The array A with elements rearranged in nondecreasing order

   for i<--1 to n-1 do
      {Insert A[i] at its proper location in A[0],A[1], . . . ,A[i-1]}
      cur<-- A[i]
      j<--i-1
      while j >= 0 and A[j] > cur do
         A[j+1] <-- A[j]
         j <-- j-1
      A[j+1] <-- cur {cur is now in the right place}
```

Code Fragment 1: Algorithmic description of the insertion-sort algorithm.

Sol:
Entries of equal value will maintain their same relative position. This is because the inner while loop stops as soon as it finds an entry of equal or smaller value.

(scratch paper)

(scratch paper)