CSE 122 Homework 2 Fall 2004

Due 11:00AM, Wednesday, September 29th, 2004

Name: ___________________________ Student ID: ___________________________

Instructions: Each question is worth 10 points. Be sure to show sufficient work to justify your answers. Partial credit will be given for partially correct answers and/or incorrect answers that show that a good amount of effort was put forth in trying to arrive at the correct answer, as long as an explanation of the answer is provided (i.e. so if you are unsure, don't just write down your answer, right down how you arrived at it and why you think it is/isn't the correct answer). There is a 20% deduction for late homework. The deduction becomes 50% if the homework is two days late. No credit is given after three days. If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. You are to work individually, and all work should be your own.

Questions:

1. When you create a class in C++, what special functions does a compiler automatically generate for you? When is it important to manually override these functions?

2. What are template classes? How are they useful, and how are they handled at compile-time (i.e. why does "code bloat" occur)?

3. List the important differences between a C style struct and a C++ style class.

4. Order the following common growth rates from slowest growth (most desirable) to fastest growth (least desirable):

   $\log^2 N$, 2^N, c, N^3, $N \log N$

5. Will an algorithm that is $O(N^3)$ **always** perform worse than one that is $O(N)$? Justify your response.

6. What are some of the most common applications of lists, stacks, and queues?

7. A stack object may be implemented using an array or by using a linked-list (among other things). What are some of the pros and cons associated with each approach?

8. What are the advantages of using pointers to store data entries in a linked list structure over simply using a contiguous array to store the data?

9. Construct a recursive and a non-recursive function that will compute the factorial of a given integer.

Recall that the factorial of n (denoted “n!”) is:

n * (n – 1) * (n – 2) * … * (n – (n – 2)) * 1, and 1! = 1

Note that this is logically equivalent to:

n * (n – 1)!

10. Compute the Order (in big-O notation) of the following code fragments:

a.
```
int j = 100;
int m = 0;

while (m < j)
{
     m++;
     cout << m <<endl;
}

do
{
     m--;
}while(m > 0);

for (int i = 0; i < j; i++)
{
     for (int k = 0; k < i;  k++)
     {
               cout << k << ”,” << i << endl;
     }
     cout << i << endl;
}
```

b.
```
int x = 7;
```

c.
```
int* binarySearch(int* array, int maxIndex, int searchTerm)
{
     //”array” must be sorted in ascending order
     if (maxIndex == 0)
     {
               if (array[0] == searchTerm)
               {
                         return &array[0];
               }
               else
               {
                         return NULL;
               }
```

```
    }
    else
    {
        int middleTerm = array[maxIndex / 2]
        if (middleTerm == searchTerm)
        {
            return &array[maxIndex / 2];
        }
        else if (middleTerm < searchTerm)
        {
            return binarySearch(&array[maxIndex / 2],
                                maxIndex / 2, searchTerm);
        }
        else //middleTerm > searchTerm
        {
            return binarySearch(&array[0], maxIndex / 2,
                                searchTerm);
        }
    }
}
```