1. Answer all questions on the Big-O review sheet.

2. The **Binary Search** algorithm follows this idea:
   
   (a) We are given a *sorted* array, and want to know if the array contains a value $x$
   
   (b) If the array is empty, return FALSE. Otherwise continue.
   
   (c) Check to see if the center element is $x$; if yes, return TRUE. Otherwise, compare $x$ to the middle element; if $x$ is larger, look in the right half. Otherwise, look in the left half.

   Answer the following:
   
   (a) Argue why this algorithm is correct.
   
   (b) What is some pseudocode that describes binary search? Give both recursive and non-recursive formulations.
   
   (c) What is the maximum number of comparisons that a binary search function will make when searching for a value in a 1,000-element array?
   
   (d) What is the maximum number of comparisons if we used a linear search instead?

3. What do LIFO and FIFO mean? Which one applies to a queue and which one applies to a stack?

4. Suppose that **MyStack** is a class in a program, and that it has implemented the stack functionality of “pop” and “push” in member functions with prototypes `public void pop(int &x)` and `public void push(int x)`. What is the output of the following code?

   ```c
   MyStack s;
   int x;
   s.push(10);
   s.push(15);
   s.pop(x);
   printf("%d", x);
   s.push(x);
   s.push(x + 4);
   s.pop(x);
   s.pop(x);
   printf("%d", x);
   ```

5. What is the asymptotic running time analysis of the following algorithm?
function f(n):
    s := 0
    for i = 1 to n
        for j = 1 to 10
            s := s + 1
        end
    end
end

6. Consider the following algorithm:

function f(n):
    if n <= 10
        return 50
    end

    s := 100
    for i = 1 to 4
        s := f(n/2)
    end

    return s

(a) What is the base case of the recursive algorithm?
(b) Write a recurrence relation that describes the running time of the algorithm.
(c) Simplify the recurrence by “unrolling” and state the running time in Big-O notation.

7. What is the output of the following program:

#include <iostream>
using namespace std;

int function(int);

int main() {
    int x = 10;
    cout << function(x) << endl;
    return 0;
}

int function(int num) {
    if (num <= 0)
        return 0;
    else
        return function(num - 1) + num;
}

8. Write a recursive function to return the number of times a specified number occurs in an array. Write
the running time of the algorithm as a recurrence relation and solve by unrolling.
9. Write a recursive function to determine if a given string (given as type `char*` along with an integer length) is a palindrome. Write the running time of the algorithm as a recurrence relation and solve by unrolling.

10. What is the best and worse case asymptotic (Big-O) running time for Selection Sort, Insertion Sort, and Bubble Sort?

11. Write the full implementation of a function that fits the prototype `bool contains(ListNode* start, int target)` that returns `true` if the target integer `x` is contained in the list that starts at node `start` and `false` otherwise. Write the function recursively, do not use any loop constructs. Assume the `ListNode` struct is defined as

   ```
   struct ListNode{
       int value;
       ListNode* next;
   }
   ```