Problem 1

a) Please find cpp file attached for stack implementation.
The time complexity for the operations of the stack are as follows:

1. Push:
O(1). This is because adding a new item to the stack is constant regardless of the current position in the stack.
2. Pop:
O(1). Popping an element from the stack takes constant time regardless of the current position in the stack.
3. isEmpty:
O(1). This operation checks if the stack is empty by checking if the pointer to the top of the stack is NULL, which takes constant time.

b) In order to implement a queue using two stacks, we need to internally reverse the order of the elements in the first stack. By doing so we will change the LIFO quality of the single stack into a FIFO data structure using the second stack. The psuedo code is illustrated below:

1.
```
enqueue(x)
        s1.push(x) //Pushing elements onto the first stack
```

2.
```
dequeue()
if(s1.isEmpty)//Check if s1 is empty
        while(!s1.isEmpty) do
                //Transfering all elements from s1 to s2
                s2.push(s1.pop);
return s2.pop;
//s2 is now s1 in reverse order
```

3.
```
isEmpty()
if(s1 and s2 == NULL) //If both of them are empty
        return true;
else
        return false;
```

Problem 2

All algorihtms were implemented as member functions of the stack class in ”Stack.h”.

a) The reverse algorithm is implemented as a member function of the stack class from **Problem 1**. Please find the algorithm in the ”Stack” folder at the bottom of the in the ”Stack.h” file. Explanation in comments.

b) Please find the code in the folder ”BstList”, in the file ”BstList.h”. The algorithm begins on line 37. Explanation in comments.
To make the program run ”make” in terminal.

c) Please find the code in the folder ”BstList”, in the file ”BstList.h”. The algorithm begins on line 64. Explanation in comments.
To run the program run ”make” in terminal.