Assignment 3

Given Date: 06/16/2022
Due Date: 06/23/2022 11:59 pm EDT

Question 1 (Stack)

(25 points) Explain how to implement two stacks in one array A[1..n] in such a way that neither stack overflows unless the total number elements in both stacks together exceeds n. Show your work for the new PUSH,POP operations for both stack1 and stack2.
(example usage: PUSH(x,A,1): push x into stack1. POP(A,2): pop from stack2.)

Answer: The *naive approach* to solve the problem is to have a dedicated space of $\frac{n}{2}$ for each one of the stack. Thus A[1..$\frac{n}{2}$] for stack1 and A[$\frac{n}{2}$+1..n] for stack2.

But it may result in inefficient use of array space since stack1 may have space but stack2 may result in overflow.

Approach 2: It uses the entire array space. Thus if space available in A, it won't cause an overflow.

To achieve that, stack1 starts from the leftmost element and first element in stack1 is pushed at index 1.

On the other side, stack2 starts from the rightmost cornet, the first element in stack2 is pushed at index n.

Hence both stack grows or shrinks in the opposite direction. The top element index of stack1 and stack2 are denoted by *top1* and *top2* respectively.

Initialization: $top1 = 0, top2 = N + 1$

Before pushing to each stack we will check whether we have space remaining in the array or not through checking $top1 < top2 - 1$ – if this condition returns true then we have space remaining to insert a new element to either stack otherwise the array is full.

Similarly, during pop operation, for stack1 we have to check whether $top1 >= 1$ – if true then we have element on stack1 to pop. For stack2, $top2 <= N$ needs to be checked.

Algorithm 1 PUSH(x,A,1)

```
if $top1 < top2 - 1$ then
    $top1 = top1 + 1$
    A[top1] = x
else
    throw a StackFullException
end if=0
```

Algorithm 2 PUSH(x,A,2)

```
if $top1 < top2 - 1$ then
    $top2 = top2 - 1$
    A[top2] = x
else
    throw a StackFullException
end if=0
```

Algorithm 3 POP(A,1)

```
if $top1 >= 1$ then
    e = A[top1]
    $top1 = top1 - 1$
    return e
else
    throw a StackEmptyException
end if=0
```

Algorithm 4 POP(A,2)

```
if $top2 <= N$ then
    e = A[top2]
    $top2 = top2 + 1$
    return e
else
    throw a StackEmptyException
end if=0
```

Question 2 (Priority Queues and Heap)

(25 points) Assume a priority queue implemented as a **sorted list** of entries using **heap**. Draw the heap after each of the operations: insert(5), insert(7), insert(3), removeMin(), insert(1), removeMin(), removeMin()

Answer: Below, I have shown the heap after every operation.

Figure 1: insert(5)

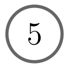


Figure 2: insert(7)

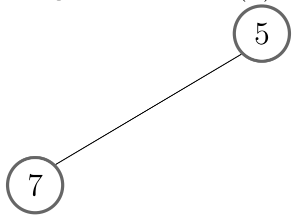


Figure 3: insert(3)

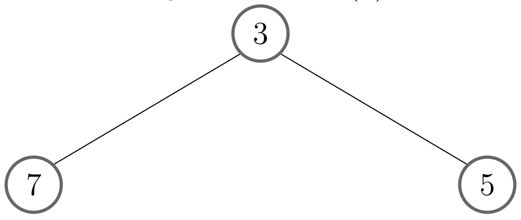


Figure 4: removeMin()

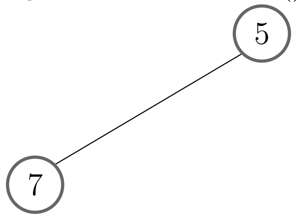


Figure 5: insert(1)

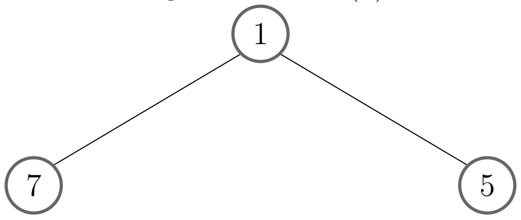


Figure 6: removeMin()

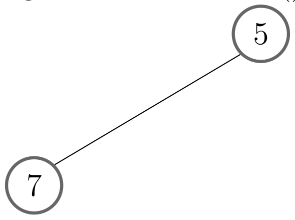
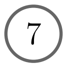
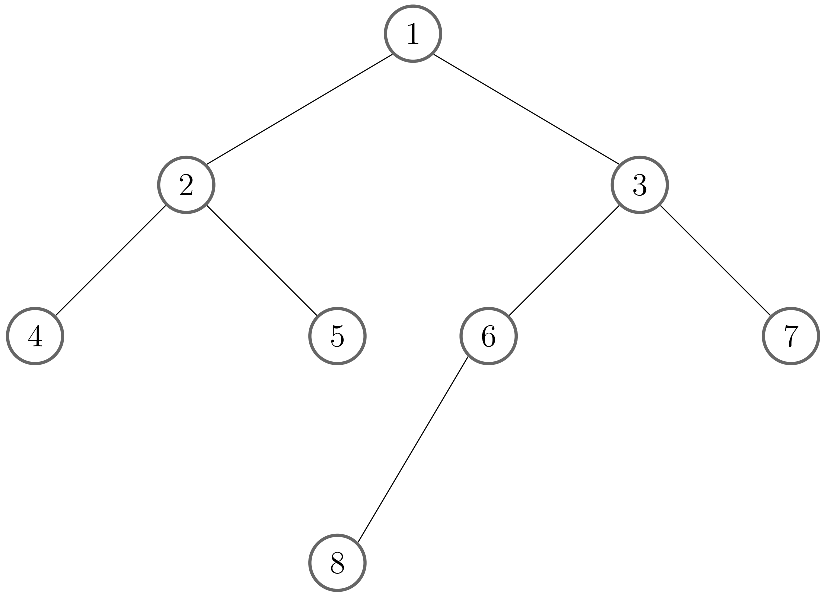


Figure 7: removeMin()



Question 3 (Trees)

1. (25 points) Consider the following tree:



Answer the following in the context of the tree above:

(a) What is the in-order traversal order of the tree?
Answer: First visit the left subtree then the root node and finally the right subtree.

The in-order traversal of the tree will be <u>**4,2,5,1,8,6,3,7**</u>

(b) What is the pre-order traversal order of the tree?

Answer: First visit the root node then traverse its children (i.e., left and right subtree).

The pre-order traversal of the tree will be <u>**1,2,4,5,3,6,8,7**</u>

(c) What is the post-order traversal order of the tree?

Answer: First traverse the children (i.e., left and right subtree) before visiting the root.

The post-order traversal of the tree will be <u>**4,5,2,8,6,7,3,1**</u>

(d) What is the height of the tree?

Answer: The Height of the tree is <u>**3**</u>.

(e) What is the depth of node 8?

Answer: The depth of node 8 is <u>**3**</u>.

(f) How many leaf nodes does the tree have?

Answer: The leaf nodes are node 4, 5, 7, 8. Hence the number of leaf nodes are <u>**4**</u>.

(g) How many internal nodes does the tree have?

Answer: The external nodes are node 1, 2, 3, 6. Hence the number of external nodes are <u>**4**</u>.

Question 4 (Graphs)

1. (25 points) Given the undirected, unweighted graph G below. Show the adjacency matrix and adjacency list representations of G.

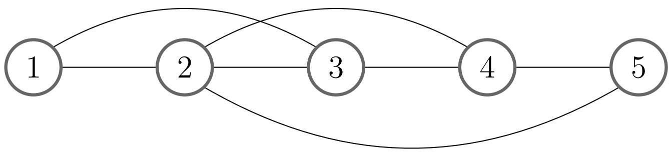


Table 1: Adjacency Matrix

	1	2	3	4	5
1	F	T	T	F	F
2	T	F	T	T	T
3	T	T	F	T	F
4	F	T	T	F	T
5	F	T	F	T	F

Figure 8: Adjacency List

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r, dash] & 3 & & \\
2 \arrow[r] & 1 \arrow[r, dash] & 3 \arrow[r, dash] & 4 \arrow[r, dash] & 5 \\
3 \arrow[r] & 1 \arrow[r, dash] & 2 \arrow[r, dash] & 4 & \\
4 \arrow[r] & 2 \arrow[r, dash] & 3 & & \\
5 \arrow[r] & 2 \arrow[r, dash] & 4 & &
\end{tikzcd}$$