CSDS 233 Due: October 7, 2022, 11:59 PM

Assignment 2

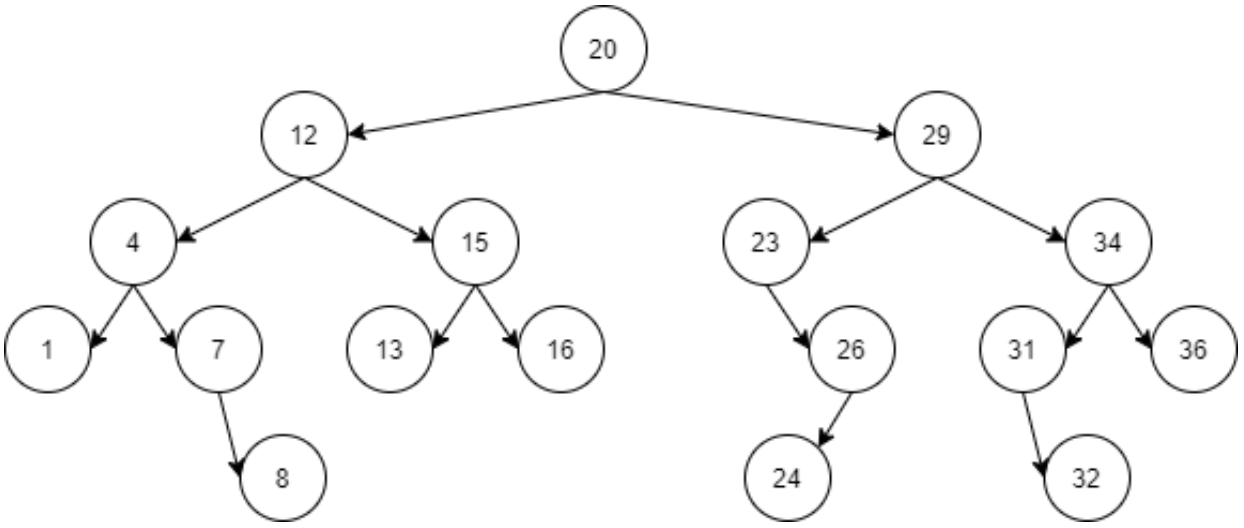
General instruction: This assignment includes two parts, written and programming. Please write/type your answers neatly so they can be readable. For the written part, please show the steps of your work. Please submit a single PDF file for the written part and a zip file for the programming part before 11:59 PM - October 7, 2022.

Special office hour for this assignment will be on Thursday Oct 6th, 2022, 6:00pm-7:00pm in Zoom (Meeting ID: 538 842 3086 Passcode: 594624), and feel free to ask question via email: rxl761@case.edu.

Written Problems (40 pts)

P.1) (20 pts) Consider the following binary tree with 18 nodes. Describe the order of nodes visited in each of the following traversals of the tree: (**4 pts each**)

a) In-Order
b) Pre-Order
c) Post-Order
d) Reverse In-Order
e) Level Order

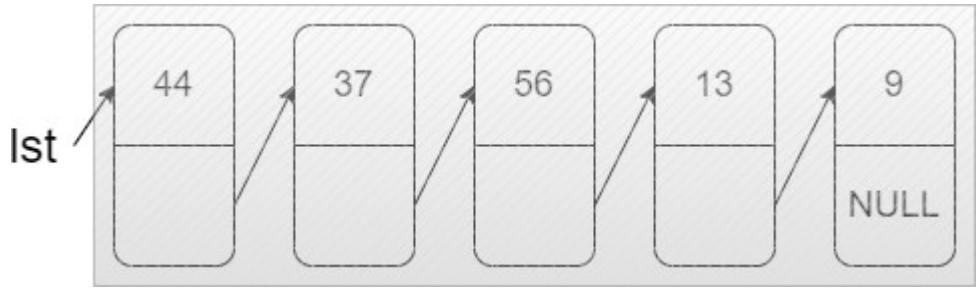


P.2) (20 pts) Convert the following expressions from infix into postfix or from postfix into infix. (**5 pts each**) (Hint: You can write $\sqrt{2}$ as 2^(1/2).)

a) -b b 2 ^ 4 a * c * - 1 2 / ^ + 2 a * /
b) A B C * E F - / + G H I K / - ^ L * +
c) $t * 1 / \sqrt{1 - v^2 / c^2}$
d) 4 / 3 * V * W ^ 3 - X * Y ^ 2 + Z

Programming Problems (60 pts)

P.1) (20 pts) Consider a singly linked list like the example given below.



Write a linear-running time method public void reverse () that would be invoked on a list object (e.g., lst.reverse()) and would reverse the list using only constant space (which basically means that you cannot use additional lists in your implementation). For simplicity, you can assume lst as a linked list of integers.

Note: This precludes using recursion since this would use a nonconstant space in the call stack memory.

P.2) (40 pts) - (20 pts each)

a) Write a class **CustomQStack** which has a single **Queue**(java.util.Queue<E>) object in its constructor. The aim of this class is to demonstrate that **Queue**s can be modified to act exactly as **Stack**s. Our **CustomQStack** must have its own methods of **empty**, **pop** and **push** methods as **Stack** class has. Luckily, **Queue** class has similar inherent methods and with slight modifications, these methods may act like those of **Stack** class'. You can assume that our custom class takes only integers as input.

b) Write a class **CustomSQueue** which has "two" **Stack**(java.util.Stack<E>) objects in its constructor. The aim of this class is to demonstrate that **Stack**s can also be modified to act exactly as **Queue**s. Our **CustomSQueue** must have its own methods of **add** and **poll** methods as **Queue** class has. Luckily, **Stack** class has similar inherent methods and with slight modifications, these methods may act like those of **Queue** class'. You can assume that our custom class takes only integers as input.

For further insight in P.2 check the following links:
https://docs.oracle.com/javase/8/docs/api/java/util/Stack.html
https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html

Example Outputs:

(These are not exact codes, they may appear differently as long as they are comprehensible)

P.1)

```
System.out.println(lst);
>> [5,6,7,8,9]
lst.reverse();
System.out.println(lst);
>> [9,8,7,6,5]
```

P.2a)

```
Queue e;
System.out.println(e);
>> [5,6,7,8,9]
CustomQStack d = new customQStack(e);
x = d.pop();
System.out.println(x);
>> 9
x = d.pop();
System.out.println(x);
>> 8
System.out.println(d.empty());
>> false
d.push(1);
System.out.println(d);
>> [5,6,7,1]
x = d.pop();
System.out.println(x);
>> 1
```

P.2b)

```
Stack e;
System.out.println(e);
>> [5,6,7,8,9]
CustomSQueue d = new customSQueue(e);
x = d.poll();
System.out.println(x);
>> 5
x = d.poll();
System.out.println(x);
>> 6
d.add(2);
System.out.println(d);
>> [7,8,9,2]
```

Submission

The submissions will be evaluated on completeness, correctness, and clarity. Please provide sufficient comments in your source code to help the TAs read it. Please generate a single zip file containing all your *.java files needed for this assignment and optionally a README.txt file with an explanation about added classes and extra changes you may have done. Name your file 'P2_YourCaseID_YourLastName.zip'. Submit your zip file electronically to Canvas. Notice: Before your programming assignment is graded, TAs will feed other examples to your code as input, so please make sure your program works for any input before your submission. You can test your code by feeding multiple inputs and check them if they are correct.

Grading (for each programming question):

- **Implementation: 6 pts**
- **Customized Demo with all functions included: 8 pts**
- **Proper encapsulation/information hiding: 2 pts**
- **Design and style: 4 pts**
    - **Style: 1 pt**
    - **Comments: 2 pt**
    - **Design: 1 pt**