Data Structures and Fundamentals of Programming (Questions 5-8)

Problem 5

Implement a function in C++ to convert postfix to fully parenthesized expressions. The process is similar to evaluation, but, instead of evaluating the expression, we form the output string, by concatenating the first operand, the operator and the second operand and placing parenthesis at the beginning and the end. The binary operators +, -, *, / are to be supported. You do not need to support unary operators. You can assume that input operands are the letters a-z. The postfix expression can be stored as a simple null terminating char-string such as below or as a C++ string. Additionally, you can assume that a generic class List, Stack, and Queue are defined as normal and you may also use the C++ string class. You can also assume that there is a function to convert a string of digits to a number and vice versa. Your function should return a string.

```
char postfix1[] = "abcde++++";
string postfix2 = "abc+d*-";
```

The first string would output "(a+(b+(c+(d+e))))", the second would output "(a-((b+c)*d))".

Problem 6

In C++ implement a generic (template) singly-linked-list class, called `Stack<item>`, that uses dynamic memory allocation. This should implement the stack ADT. The stack should look something like the following:

`TOS -> X1 ->X2 -> … -> Xn -> NULL`

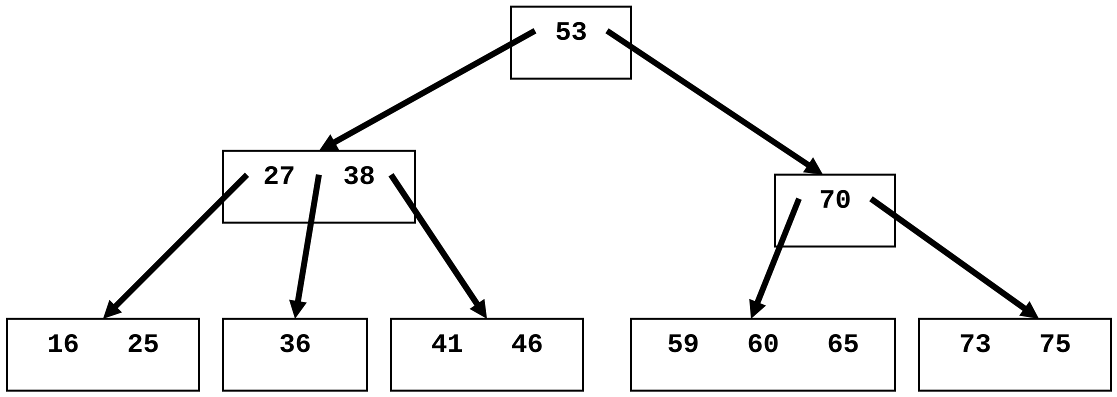
where X_1 is the node on the top of the stack and X_n is at the bottom of the stack. Besides `Stack`, you will most likely want another class or struct called `node`. Along with the class definition(s), you must implement the following methods for `Stack<item>`:

- Default constructor
- Destructor
- Copy-constructor
- Push – adds an item to the stack
- Pop – removes and item from the stack

Note: Your implementation can **NOT** use STL or any other libraries (standard or otherwise).

Problem 7

For the following 2-3-4 tree give the corresponding red-black tree.



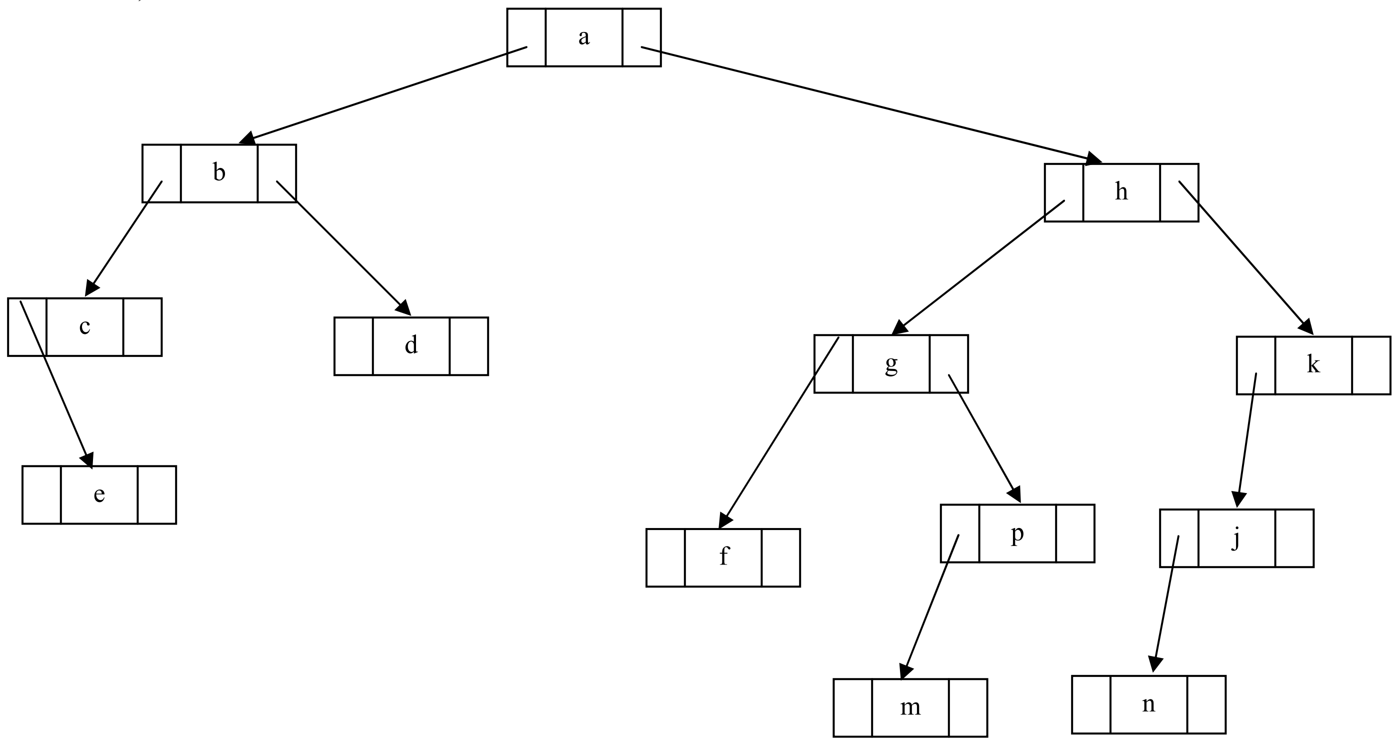
Problem 8

A) Convert the following infix expressions into postfix and prefix.

`a * b * c * d - e * d – f - g`

`a * (b + c) * (d - e) – d * f`

B) Give the Preorder, Postorder, and Inorder traversals of the tree below:


a
b
h
c
d
g
k
e
f
p
j
m
n