BIRZEIT UNIVERSITY

COMP2421 - Data Structures

Date: Wednesday 8/12/2021 | Midterm Exam - Fall 2021/2022 | Time: 90 minutes

Student Name: _____ Student ID: _____

Question #1 [20 pts]

1. What is the running time of the following code fragment? Explain your answer.

```
for( i = 0; i < n; i++ )
        for( j = 0; j < i *  i; j++ ){
        }
```

$-n$ 3

$-n^2$ 3

$O(n^3)$

Final 4

2. Given the following function, write its running time as function of Big O.

```
int Func(int A[], int n) {
        if(n<= 1)
            return 0;
        else
            return Func(A,n/2) + 6 * Func(A,n/2);
}
```

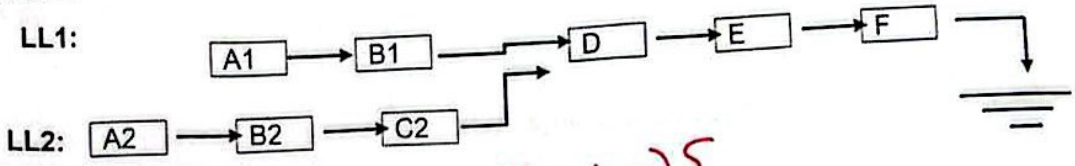
Relation = 4

Sol. = 3

Final answer = 2

Question #2 [20 pts]

Assume you have two linked lists that share some nodes, write a function that finds the first common node.




```
ptr common( ptr L1 , ptr L2){
  ptr a=L1->next;
  ptr b;
  while(a!=NULL){
    b=L2->next;
    while(b!=Null){
      if(a==b)
        return a;
      else
        b=b->next;
    }
    a=a->next;
  }
  return NULL;
}
```

Question #4 [20 pts]

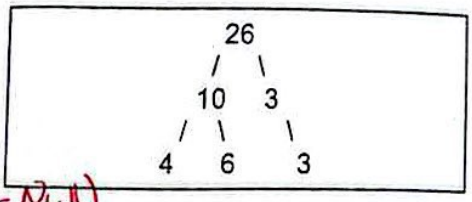
Write a function that returns **1** if a given Binary Tree is **SumTree** else returns **0**.

A **SumTree** is a Binary Tree where the value of a node is equal to the sum of all nodes in its left subtree and right subtree.

An empty tree is SumTree and the sum of an empty tree can be considered as 0.

A leaf node is also considered as SumTree.

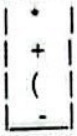
The following is an example of SumTree.



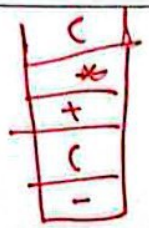
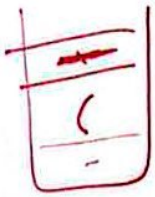
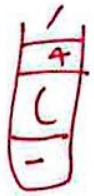

```
int Sum(BT T){
  if(T == Null)
    return 0;
  if(T->left == Null && T->Right == Null)
    return T->data;
  else{
    return Sum(T->left) + T->data + Sum(T->Right);
    }
}

int SumTree(BT T){
  if(T == Null || (T->Left == Null && T->Right == Null))
   return 1;
  if( SumTree(T->left) && SumTree(T->Right)){
     if( sum(T->left) + Sum(T->Right) = T->data)
       return 1;
      else
        return 0;
  }

  return 0;

}
```

Question #3 [20 pts]

A) Consider the usual algorithm to convert an infix expression to a postfix expression. Suppose that you have read the i^{th} input characters during a conversion and that the stack now contains these symbols:



Now, suppose that you read and process the $(i+1)^{th}$ symbol of the input. Draw the stack for each separate case where the symbol is:

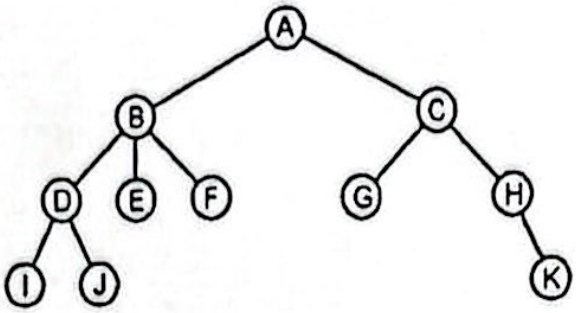
A left parenthesis: (	A minus sign: -	A division sign: /
		

B) Write a function that takes two sorted stacks R and Y (min on top), then create another stack D that merges the two stacks and should be also sorted (min on top).
You are allowed to use only the stack operations such as pop, push, size and top. No other data structure such as arrays are allowed. You are allowed to use stacks only.

```
Stack Merge(Stack R, Stack Y){
  Stack D; C;
int a = R.Top;
int b = Y.Top;
while(!isEmpty(R) && !isEmpty(Y)){
    if(a<b){
      push(C,a)
      Pop(R);
    }
    else if(b<a){
      Push(C,b);
      Pop(Y);
    }
    else {
      Push(C,a)
      Pop(R);
      Pop(Y);
    }
    a = R.Top;
    b = Y.Top;
}
while(!isEmpty(R)){
    a = R.Top;
    Push(C,a);
    Pop(R);
}
while(!isEmpty(Y)){
    b = Y.Top
    Push(C,b)
    Pop(Y);
}
while(!isEmpty(C)
    push(D,C.Top
    Pop(C);
}
return D;
}
```

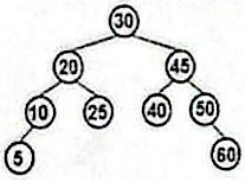
Question #5 [20 pts]

A) Assume you have the tree below; write its **post order** traversal. [10]

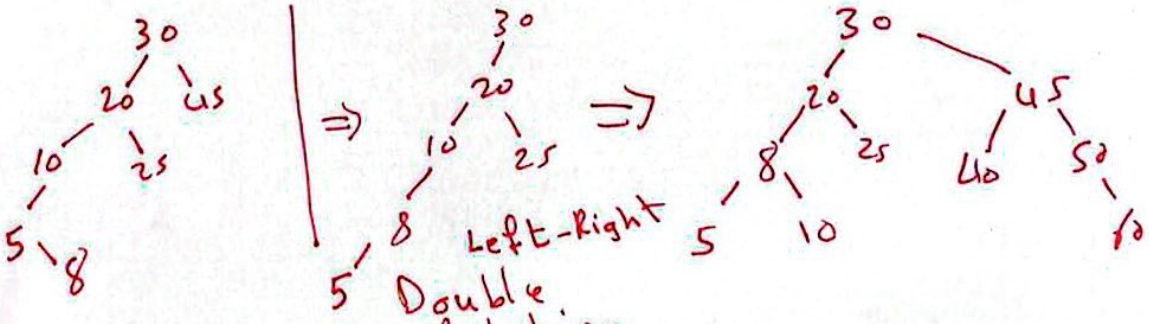


Post order	I	J	D	E	F	B	G	K	H	C	A

B) Consider the following AVL tree below.[10]



i. Insert a new node with a value of **8** into an AVL tree, show all steps.



ii. Delete the root value from the tree you have obtained from the previous step, show all steps.