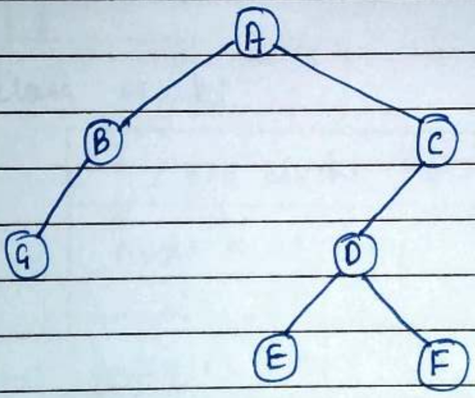
* Binary Tree

- A tree is binary if each node of the tree can have maximum of two children.
- moreover, children of a node of binary tree are ordered.
- One child is called the 'Left' child and the other is called the 'right' child.
- An example of binary tree is shown in the figure.
- Node A has two children B and C.
- Similarly, nodes B and C, ~~a~~ each have one child name G and D respectively.



udocode for struct node

struct node
char data node *left node *right

→ Pseudocode for class tree

class tree
char prefix[20] node *top
void expression(char[]) void display(node *) void non_rec_postorder(node *) void del(node *)

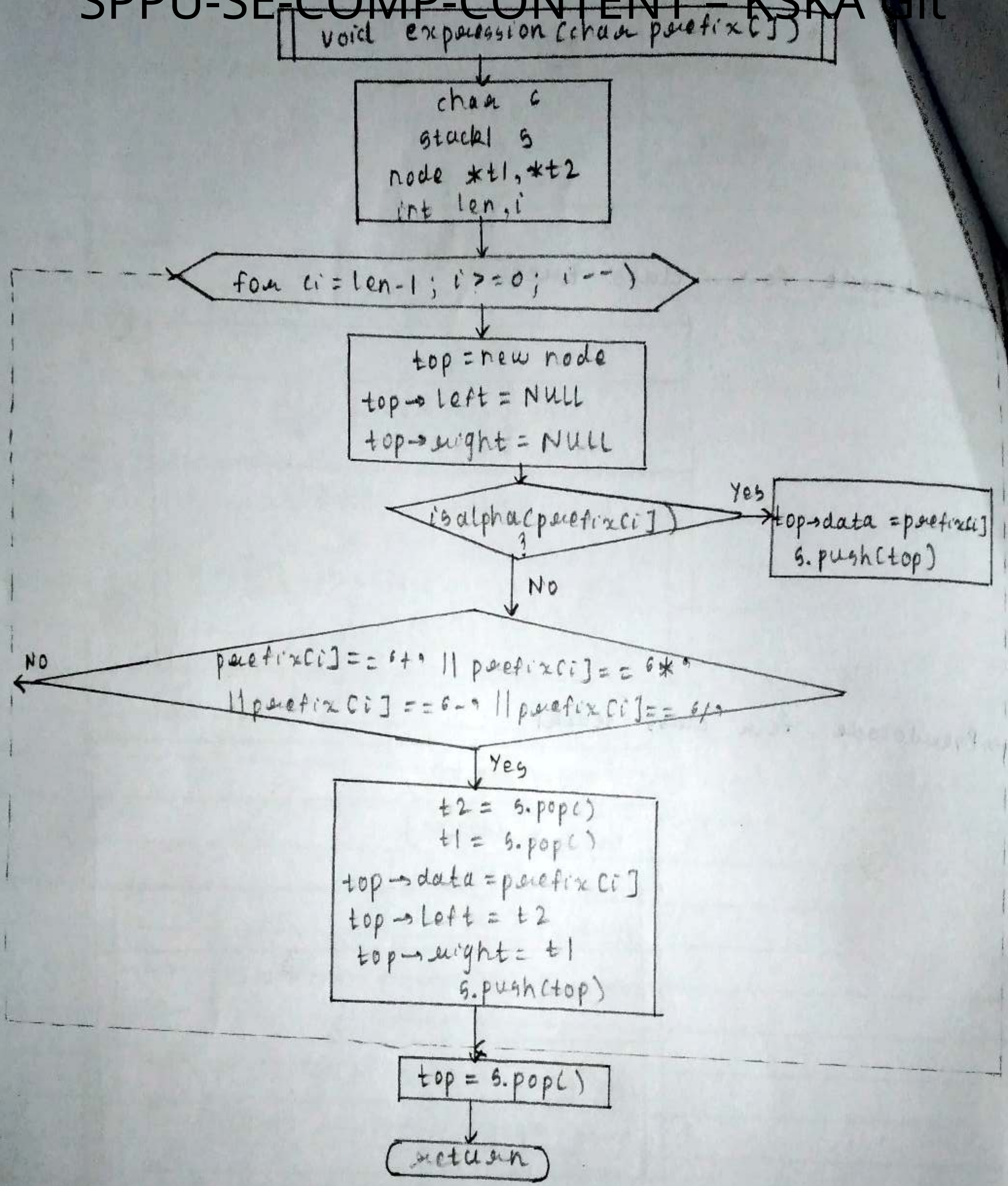
→ Pseudocode for class stack1

class stack1
node * data[30] int top
stack1() { top=-1; }

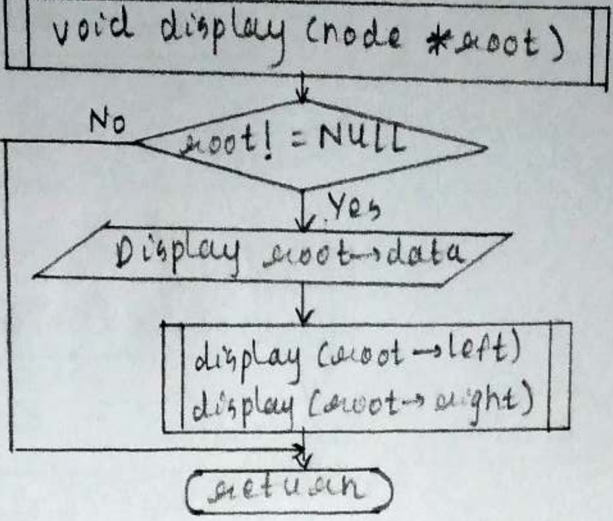
int empty{ if (top==-1) return 1;
return 0; }

void push (node *p){ data[++top]=p; }
node *pop(){ return data[top--]; }

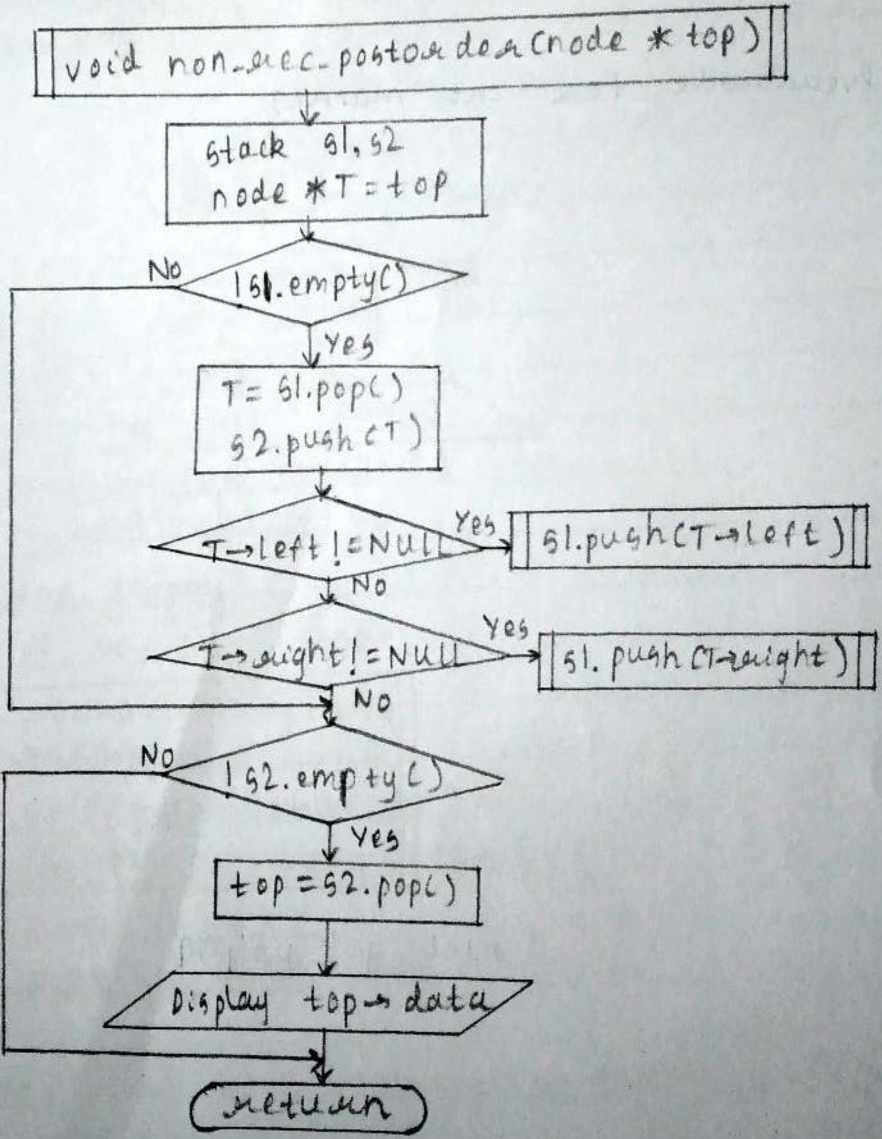
→ Pseudocode for void expression (char prefix[])



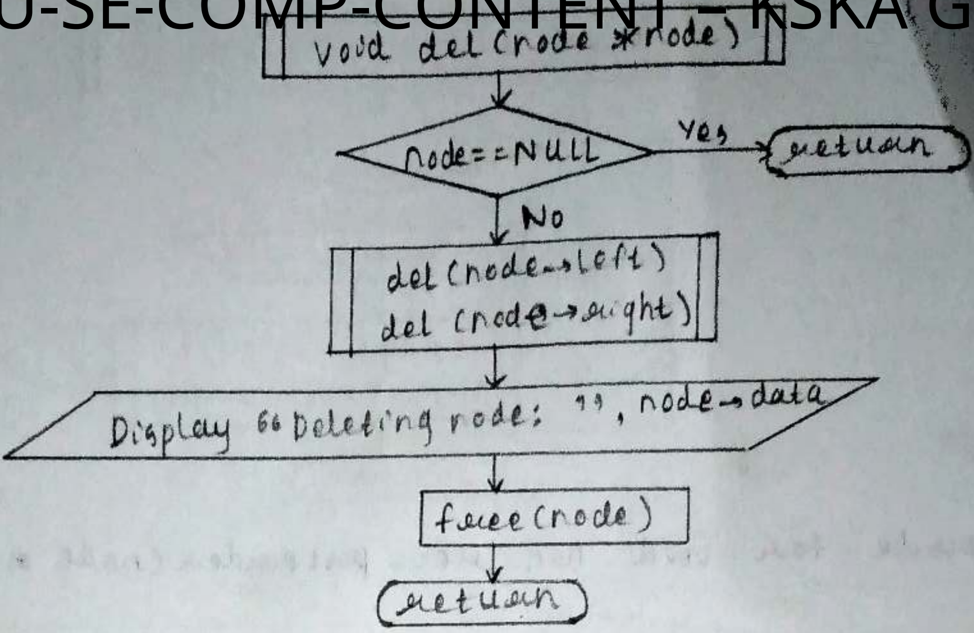
for void display (node *root)



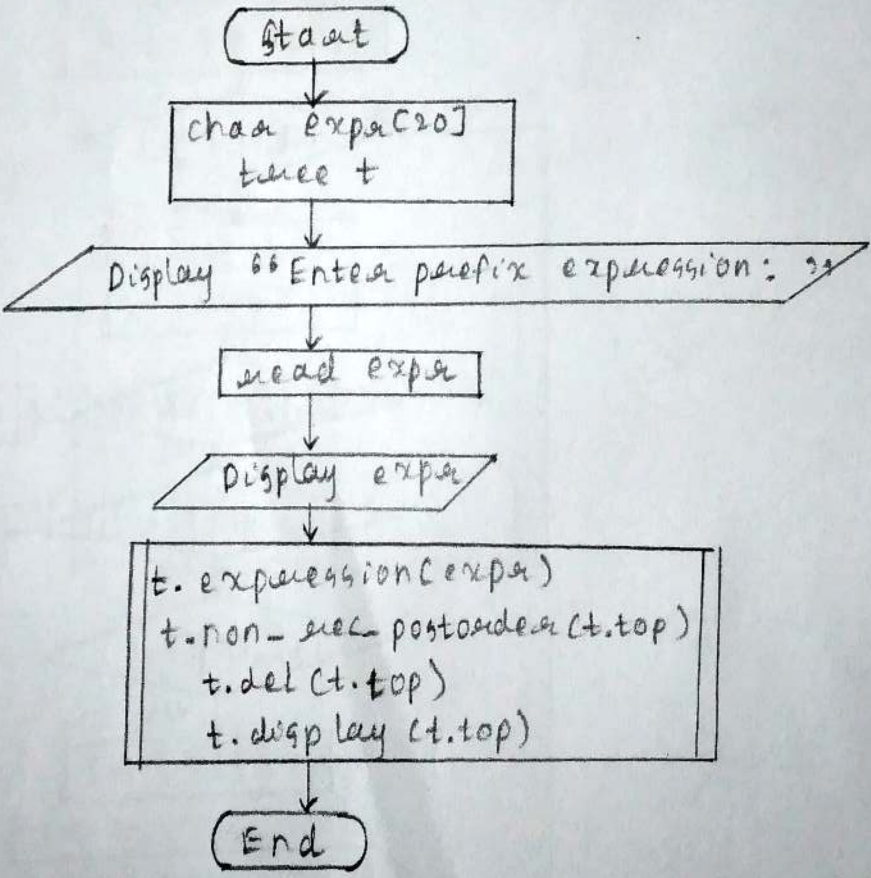
→Pseudocode for void non-rec-postorder (node * top)



→ Pseudocode for void ~~tree~~ del (node *node)



→ Pseudocode for int main()



```
→ Pseudocode for struct node
1. Declare char data
           node *left
           node *right
2. ~~set~~ return

→ Pseudocode for class tree
1. Declare char prefix [20]
2. Declare node *top
3. Create function void expression (char [])
                   void display (node *)
                   void non_rec_postorder (node *)
                   void del (node *)

→ Pseudocode for class stack1
1. Declare node *data [30]
           int top
2. create constructor stack1 ()
           initialize top = -1
3. Declare int empty ()
           if top == -1 then
                 return 1
           return 0
4. Declare void push (node *p)
           initialize data [++top] = p
5. Declare node *pop()
           return data[top--]

→ Pseudo code for void expression (char prefix[])
1. Declare char c
2. Create stack1 object s
3. Declare node *t1, *t2
```

```
4. Declare int len,i
5. initialize len= strlen(prefix)
6. for i=len-1 ; i>=0; i~~++~~ do
   begin
        initialize top= new node
        store top->left = NULL
        store top->right = NULL
        if isalpha(prefix[i]) then
              initialize top->data = prefix[i]
              s.push(top)
        else if (prefix[i] =='+' || prefix =='*' ||
              prefix[i] =='-' ||prefix == '/'). then
              initialize t2 = s.pop()
              initialize t1 = s.pop()
              store top->data= prefix[i]
              store top->left = t2
              store top->right = t1
              s.push(top)
   end
7. initialize top = s.pop()
8. return

→ Pseudocode for void ~~trace~~ display (node *root)
1. if root! =NULL then
        display root->data
        display (root->left)
        display (root-> right)
2. return

→ Pseudocode for void non_rec_postorder(node *top)
1. Create object S1,S2
2. Declare node *T = top
```

```
3. call function s1.push(T)
4. while !s1.empty() do
   begin
           store T = s1.pop()
           call function s2.push(T)
           if (T→left != NULL) then
                   call function s1.push(T→left)
           if (T→right != NULL) then
                   call function s1.push(T→right)
   end
5. while !s2.empty() do
   begin
           initialize top = s2.pop()
           display top→data
   end
6. return
```

→ Pseudocode for void del(node * node)

```
1. if node == NULL then
           return
2. call function del(node→left)
                 del(node→right)
3. Display "Deleting node: ", node→data
4. free(node)
5. return
```

→ Pseudocode for int main()

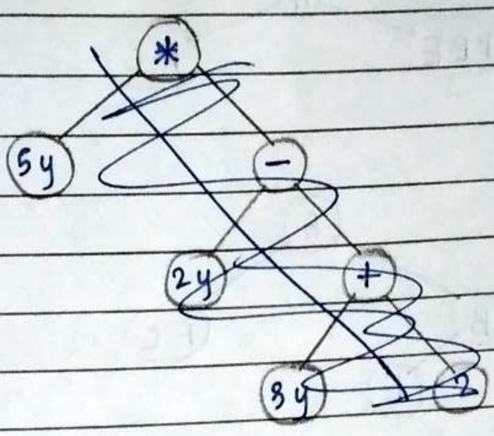
```
1. ~~Declare char expr[20~~ Start
2. Declare char expr[20]
3. ~~8~~ Create object t
4. Display "Enter prefix expression: "
5. read expr
```

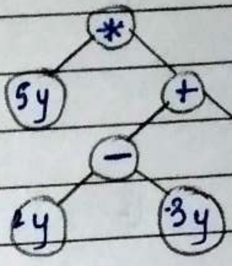
```
6   display expr
7.  call function t.expression (expr)
8.  call function t. non_rec_postorder (t.top)
9.  call function t.del (t.top)
10. call function t.display (t.top)
11. End
```

Q1. Consider the polynomial 5y(2y-3y+2)

a) Write the polynomial as an expression tree that obeys the usual ordering of operations.

Ans.

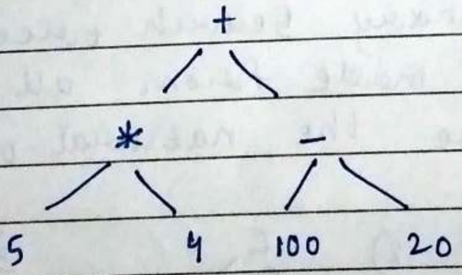




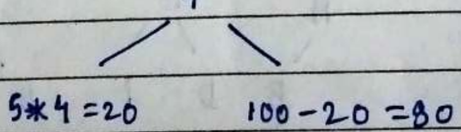
b) Write the polynomial as as postfix expression

Soln: ~~5y 2y 3y 2 + − *~~ 5y 2y 3y − 2 + *

Q2. Given a full binary tree consisting of basic binary operators (+, −, *, /) and some integers. Your task is to evaluate the expression tree



Soln:

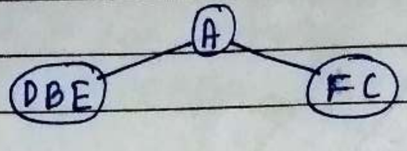


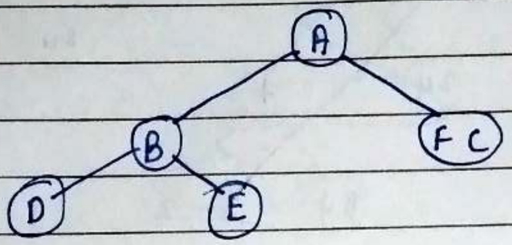
20+80 =100

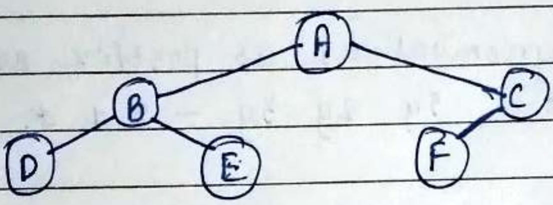
∴ Final answer =100

Q3. a) Draw the binary tree whose in-order traversal is DBEAFC and whose pre-order traversal is ABDECF.

Ans.

① 

② 

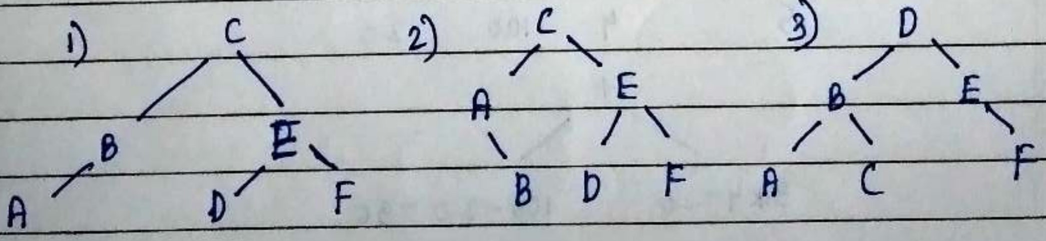
③ 

b) What is the post-order traversal of this tree?

Ans. DEBFCA

c) Draw all binary search trees of height 2 that can be made from all the letters ABC.DEF, assume the natural ordering.

Soln:


1)
C
B
E
A
D
F
2)
C
A
E
B
D
F
3)
D
B
E
A
C
F