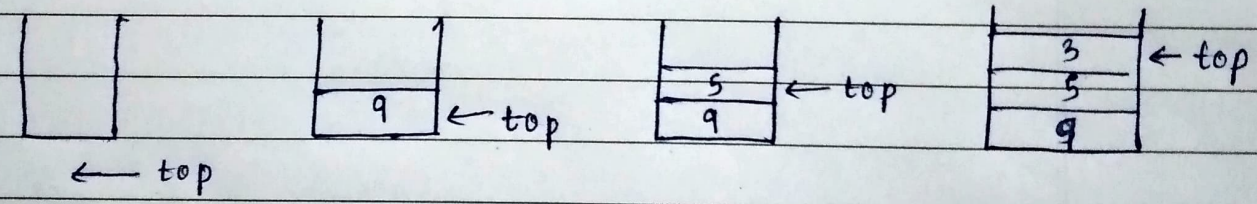
* Data structure used:-

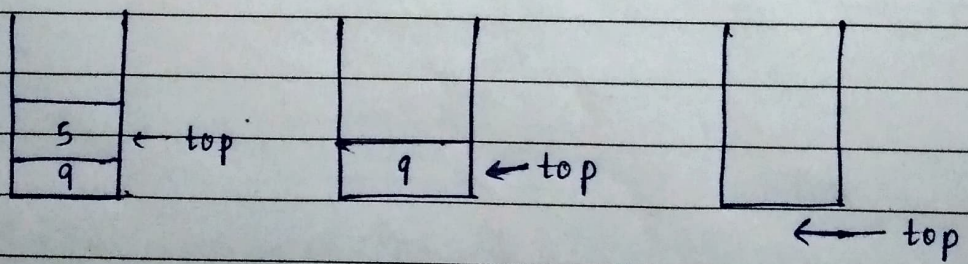
1) Stack:-

- Stack is a LIFO (last in first out) structure.
- It is an ordered list of the same type of elements.
- A stack is a linear list where all insertions and deletions are permitted only at the end of the list.
- When elements are added to stack it grows at one end.
- Similarly, when elements are deleted from a stack, it shrinks at the same end.

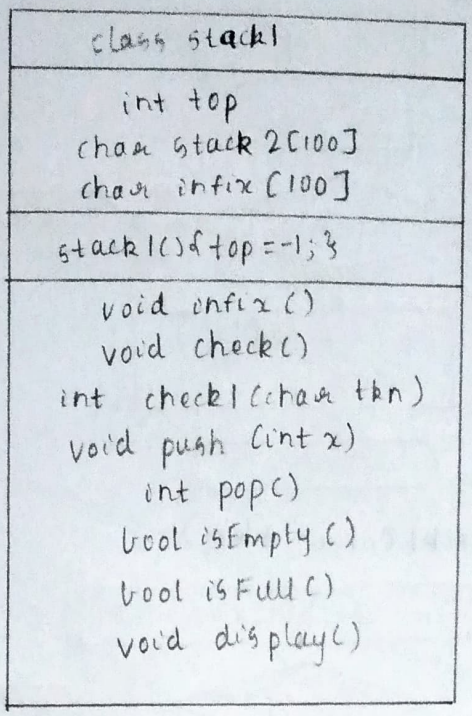
→ addition of elements:-



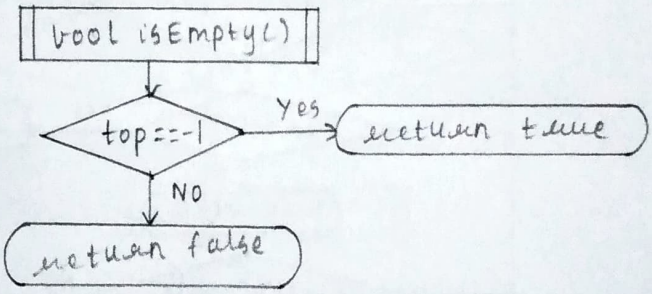
→ deletion of elements:-



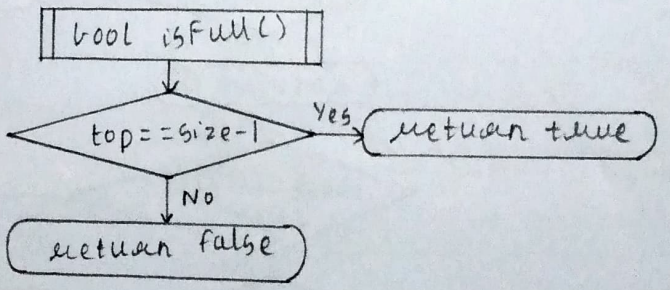
wchaart for class stack1



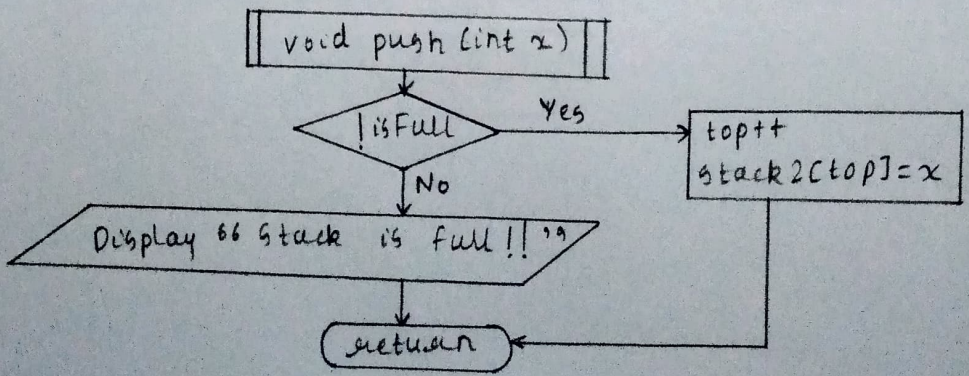
→ Flowchaart for bool stack1::isEmpty()



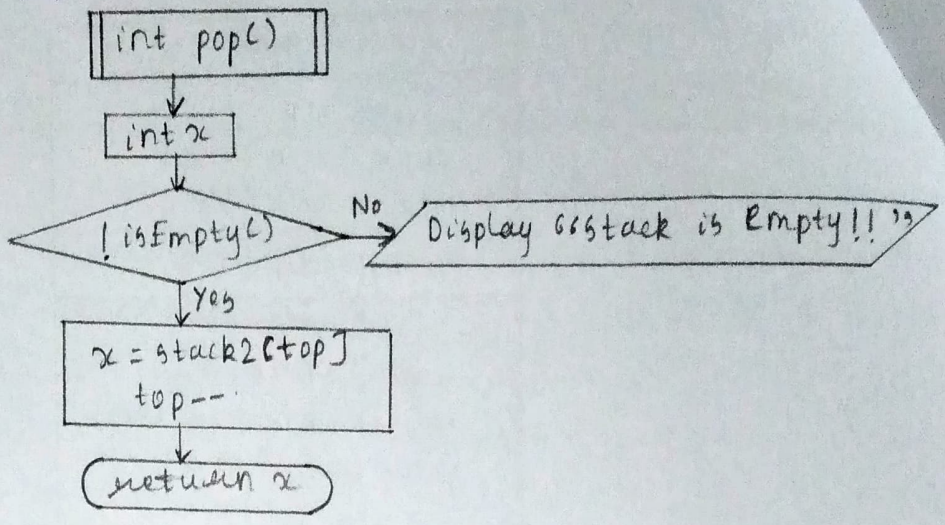
→ Flowchaart for bool isFull()



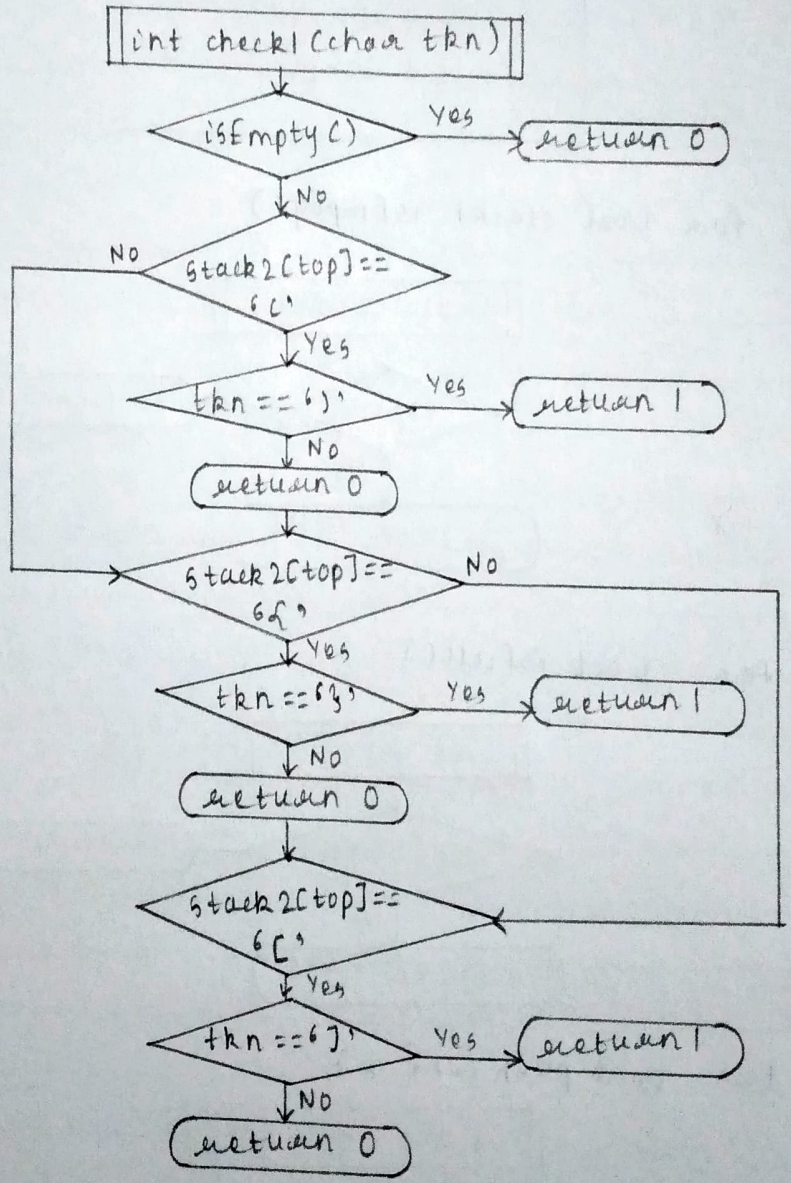
→ Flowchaart for void push(int x)



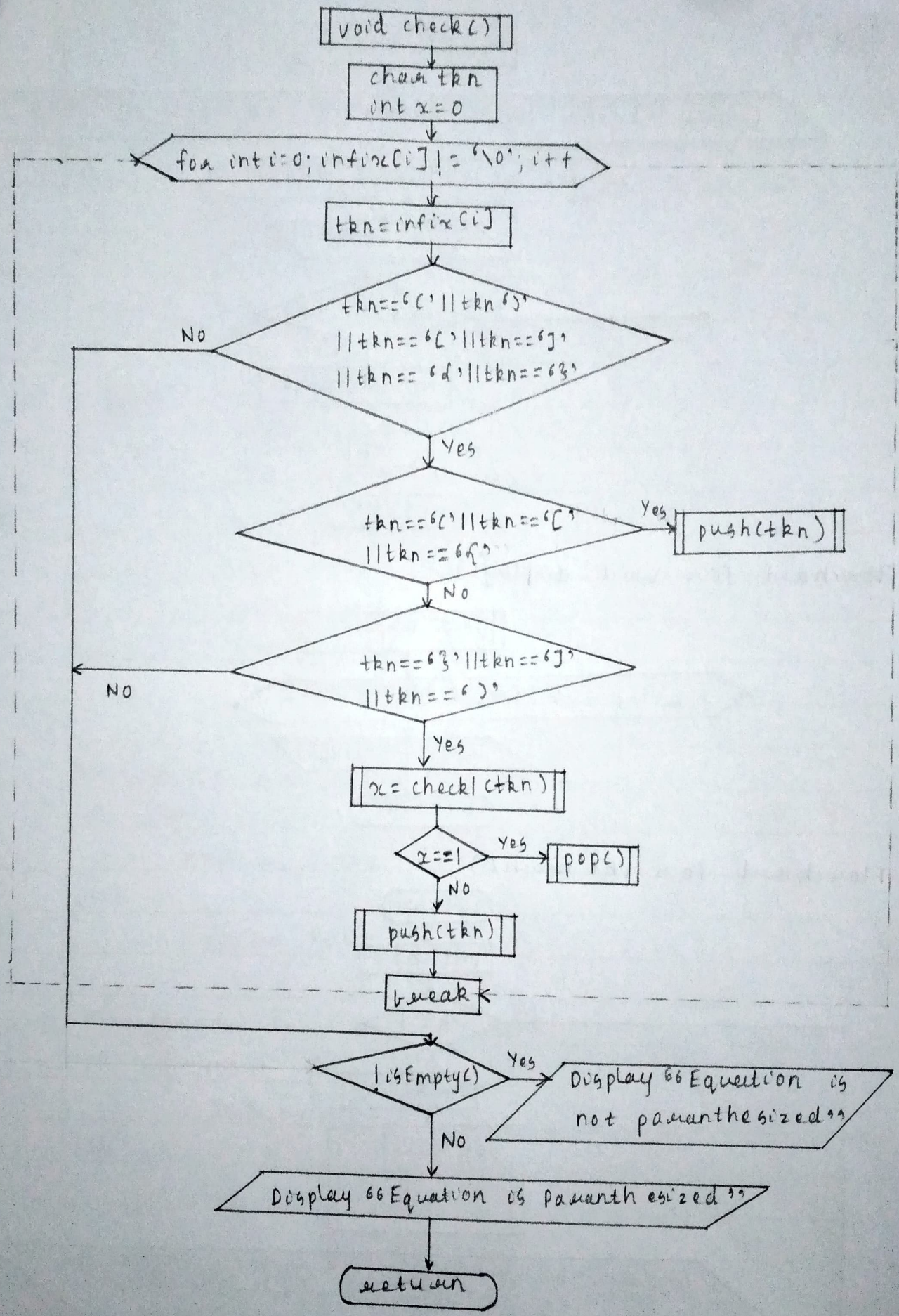
→ Flowchart for int pop()



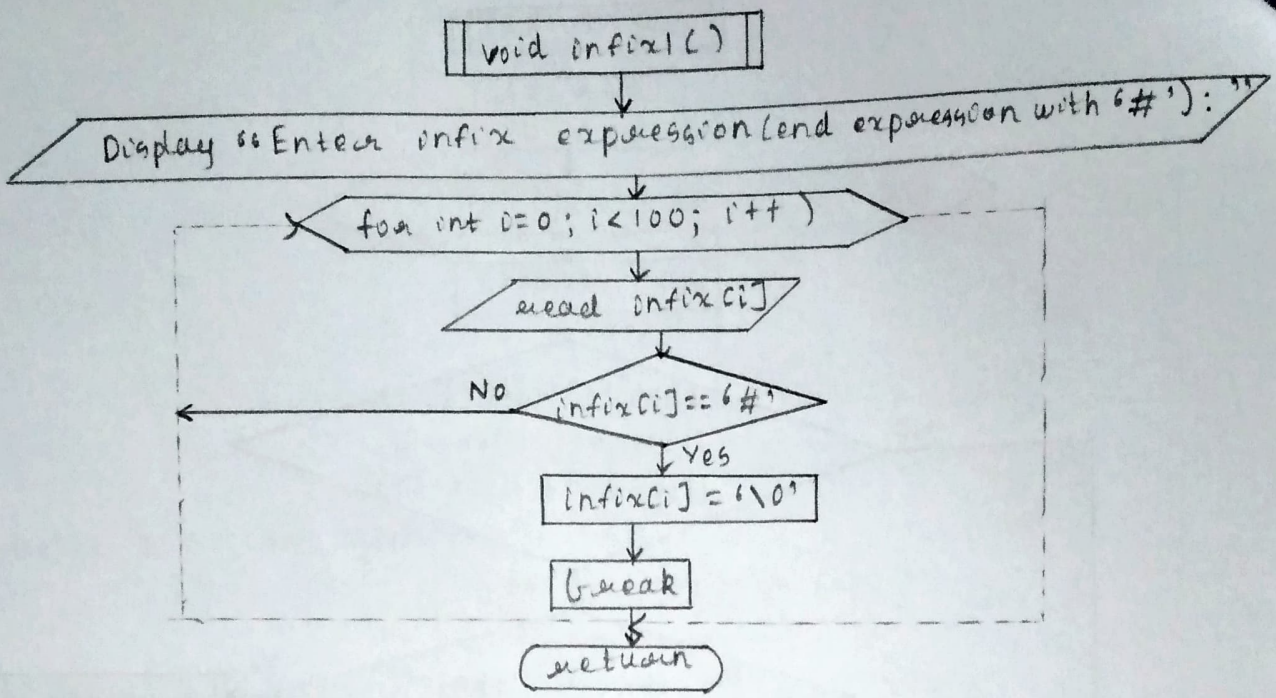
→ Flowchart for int check1(char tkn)



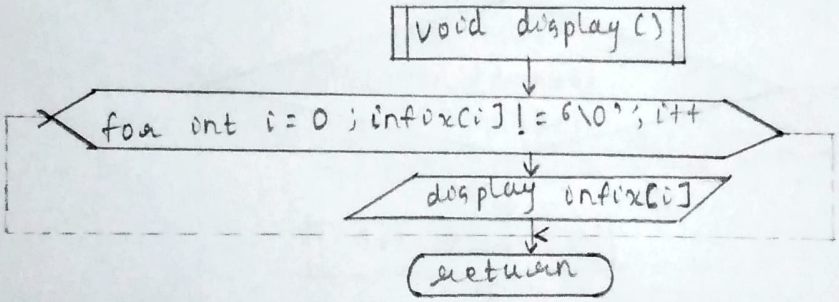
Flowchart for void check()



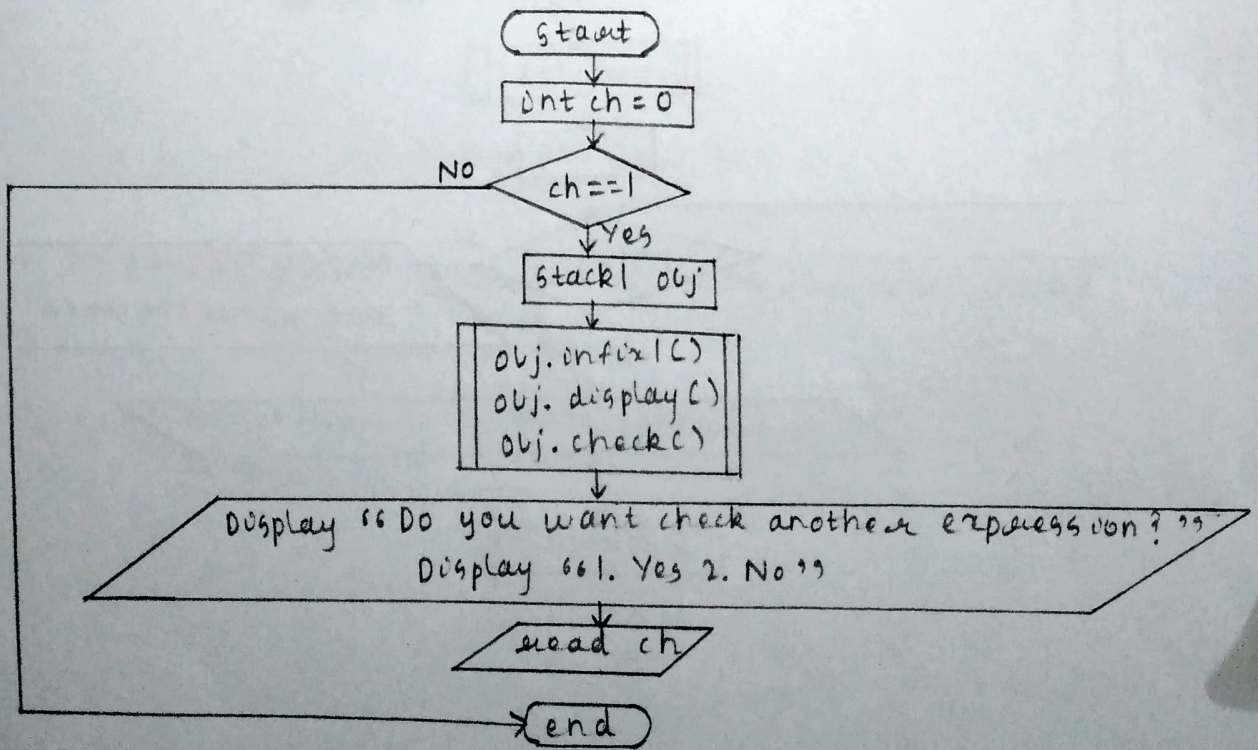
→ Flowchart for void infix1()



→ Flowchart for void display()



→ Flowchart for int main()



→ Pseudocode for class stack1

1. Declare int top
   char stack2 ~~C100~~ C100]
   char infix C100]

2. Create stack1()
   top = -1

3. Create void infix1()
   void check()
   int check1 (char ttn)
   void push (int x)
   int pop()
   bool isEmpty()
   bool isFull()
   void display()

→ Pseudocode for bool isEmpty()

1. if top == -1 then
   return true
   else
   return false

→ Pseudocode for bool isFull()

1. if top = 100 - 1 then
   return true
   else
   return false

→ Pseudocode for void push (int x)

1. if !isFull() then
   increment top
   store stack2[top] = x

else

Display "stack is full!!"

2. return

→ Pseudocode for int pop()

1. Declare int x
2. if !isEmpty() then
   store x = stack2[top]
   decrement top
   return x

else

display "stack is empty!!"

→ Pseudocode for int check1(char tkn)

1. if isEmpty() then
   return 0
2. if ~~str~~ stack2[top] == '(' then
   if tkn == ')' then
   return 1
   else
   return 0
3. if stack2[top] == '{' then
   if tkn == '}' then
   return 1
   else
   return 0
4. if stack2[top] == '[' then
   if tkn == ']' then
   return 1
   else
   return 0

→ Pseudocode for void check()

1. Declare char tkn
2. Initialize int x=0
3. for i=0; infix[i]!='\0'; i++ do

begin

store tkn=infix[i]

if tkn=='(' || tkn=='{' || tkn=='[' || tkn==']' ||
tkn=='(' || tkn=='}' then

if tkn=='(' || tkn=='[' || tkn=='(' then

call function push(tkn)

if tkn=='}' || tkn==']' || tkn==')' then

store x=check1(tkn)

if x==1 then

call function pop()

else

call function push(tkn)

break

end

4. if !isEmpty() then

Display "Equation is not paranthesized!!"

else

Display "Equation is paranthesized!!"

5. return

→ Pseudocode for void infix1()

1. Display "Enter infix expression (end expression with '#'): "
2. for i=0; i<100; i++) do

begin

read infix[i]

if infix[i] == '#' then
infix[i] = '\0'
break
end
3. return

→ Pseudocode for void display()
1. for int i=0; infix[i] != '\0'; i++
begin
display infix[i]
end
2. return

→ Pseudocode for int main()
1. Start
2. Initialize int ch=0
3 while ch==1 do
begin
create stack1 obj
call function obj.infix1()
call function obj.display()
call function obj.check()
Display "DO YOU WANT TO CHECK ANOTHER EXPRESSION?"
Display "1. YES 2. NO"
read ch
end
4. Stop

Q1 Explain the stack operations.

Ans-1) isEmpty():- checks if whether the stack is empty or not.

- It returns true if stack is empty, otherwise false.

2) isFull():-

- checks if whether the stack is full or not.
- It returns true if stack is full, otherwise false.

3) push(x):-

- Adds the element 'x' at the top of the stack

4) pop()

- Deletes the most recent element entered ~~of~~ of the stack

5) display()

- It is used to display elements present in the stack.

Q2. Write down applications of stack.

Ans. Stack data structure is very useful. Few of its usages are given below:-

1. Expression ~~evel~~ conversion
   a) Infix to postfix
   b) Infix to prefix
   c) Postfix to infix
   d) Prefix to infix
2. Expression evaluation
3. Parsing
4. Simulation of recursion
5. Function call