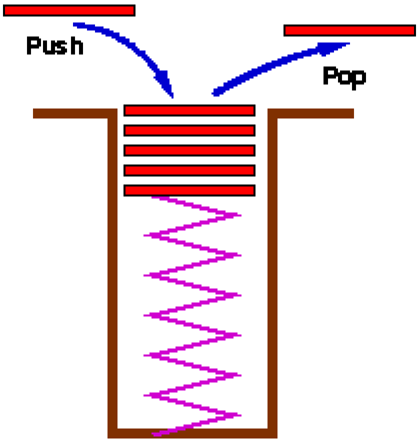
Practical no -7

Aim: Write a program to implementation of a stack using one dimensional array.
STACK-

A stack is simply a list of elements with insertions and deletions permitted at one end—called the stack top. That means that it is possible to remove elements from a stack in reverse order from the insertion of elements into the stack. Thus, a stack data structure exhibits the LIFO (last in first out) property. Push and pop are the operations that are provided for insertion of an element into the stack and the removal of an element from the stack, respectively. Shown in Figure 19.1 are the effects of push and pop operations on the stack.



A common model of a stack is a plate or coin stacker. Plates are "pushed" onto to the top and "popped" off the top.

Stacks form Last-In-First-Out (LIFO) queues and have many applications from the parsing of algebraic expressions to ...

Figure 19.1: Stack operations.

Since a stack is basically a list, it can be implemented by using an array or by using a linked representation.

Array Implementation of a Stack

When an array is used to implement a stack, the push and pop operations are realized by using the operations available on an array. The limitation of an array implementation is that the stack cannot grow and shrink dynamically as per the requirement.

Algorithm:

For push operation:

Steps:

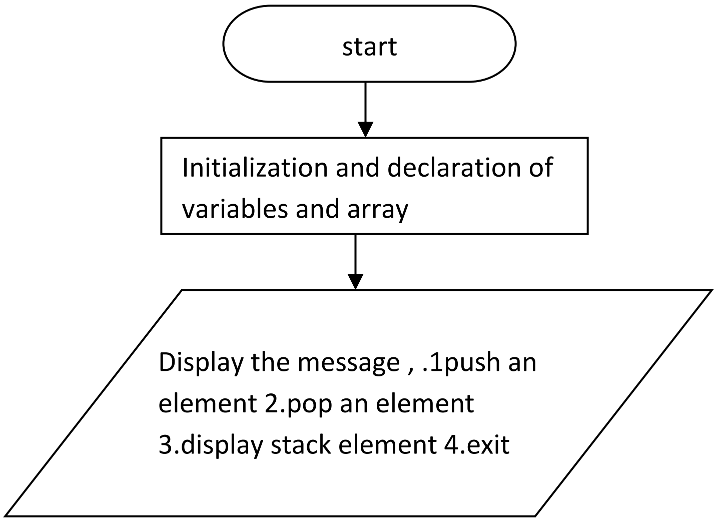
```
Push(array[],element)
1.[check for overflow]
        If  top>=size
        Printf ”Stack overflow”  and exit.
2.[increment the pointer i.e top by 1]
        top=top+1
3.[perform insertion]
        Array [top] =element.
4. Exit
```

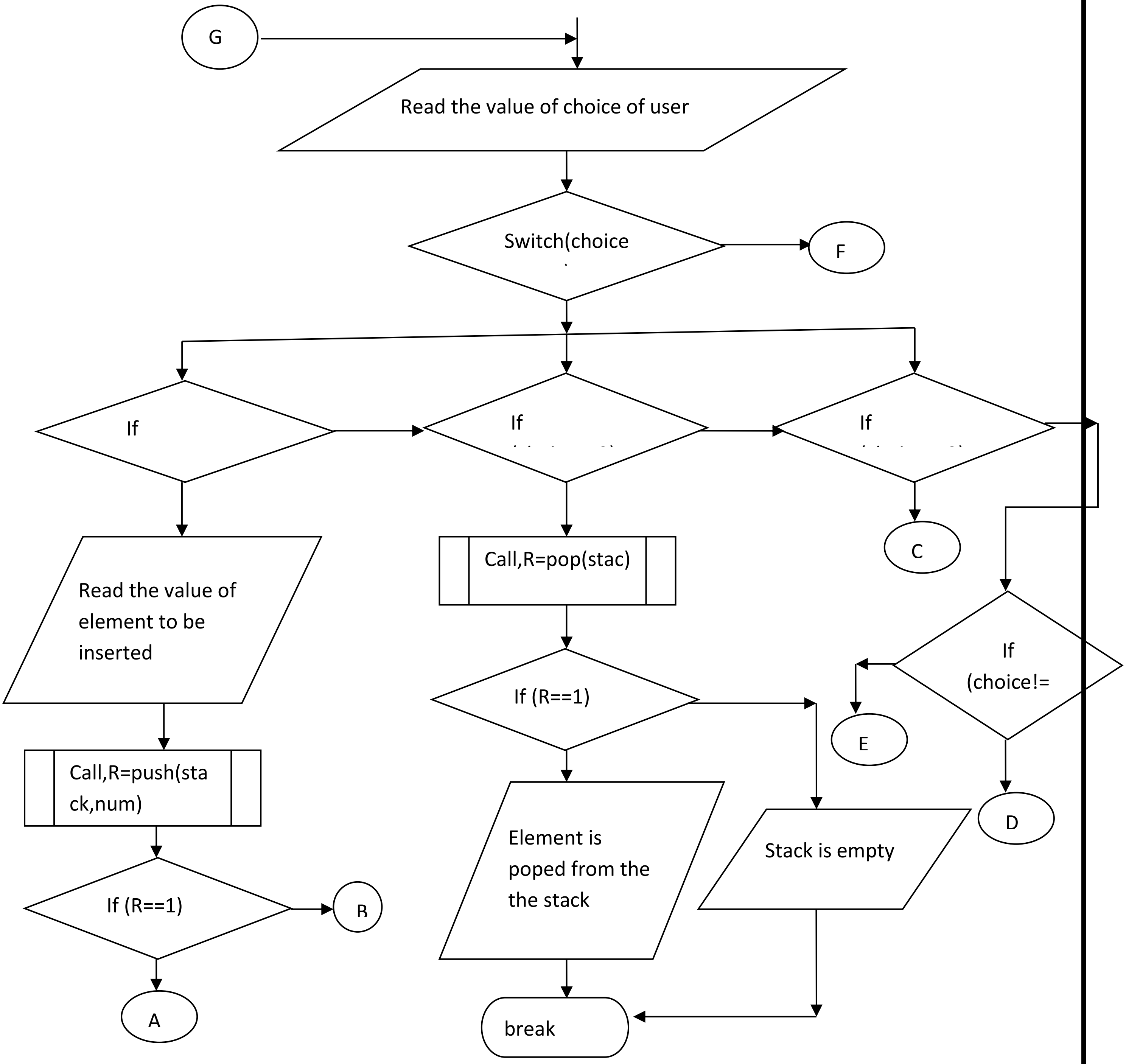
For pop operation:

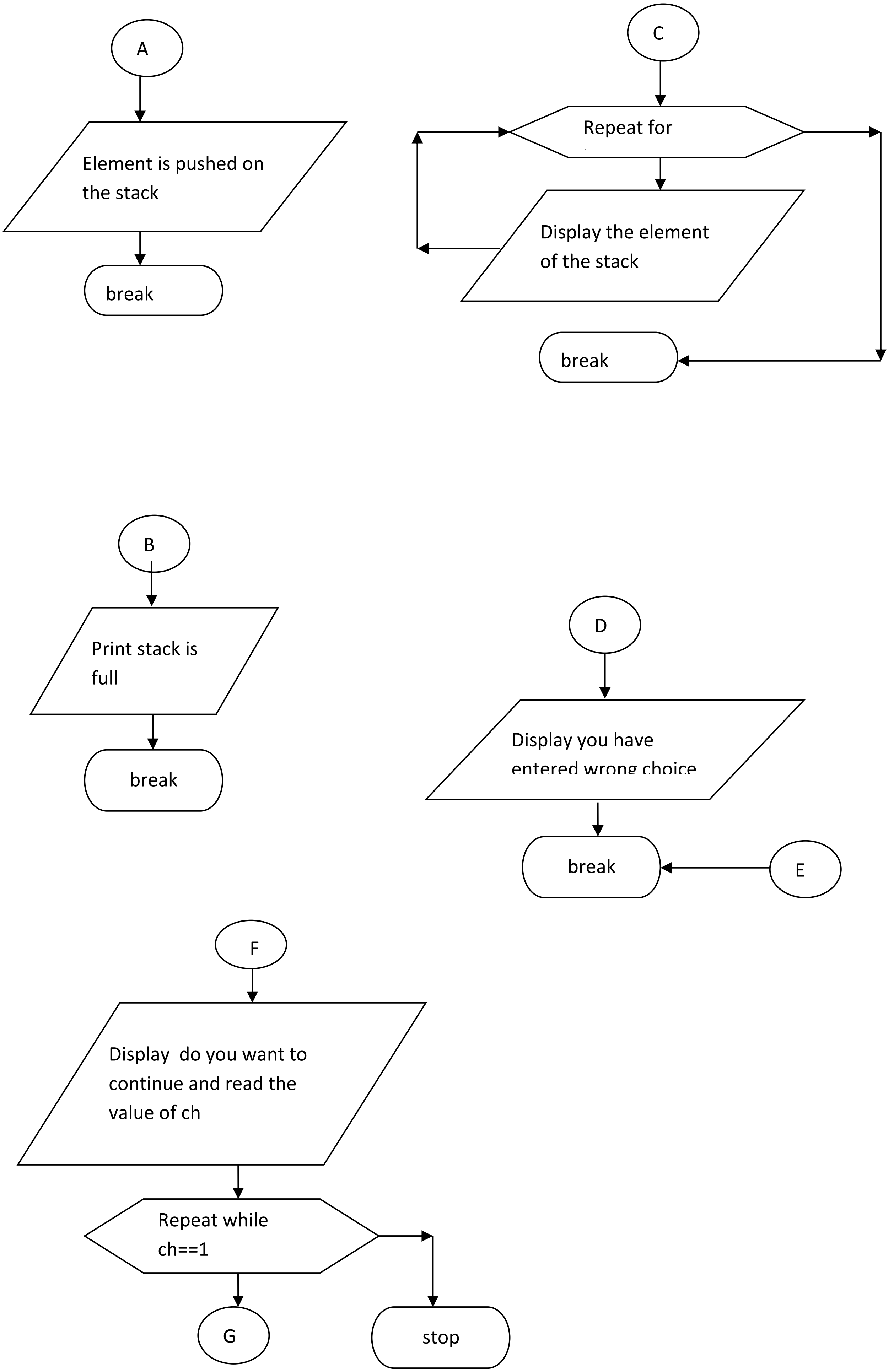
Steps:

```
Pop(array[])
1.[check for underflow]
        If top=0
        Print”Stack underflow” and exit.
2.[remove the top information]
        Element=array [top]
        Top=top-1
3. [Return the information of the stack]
        Return (element).
4.Exit
```

Flowchart:

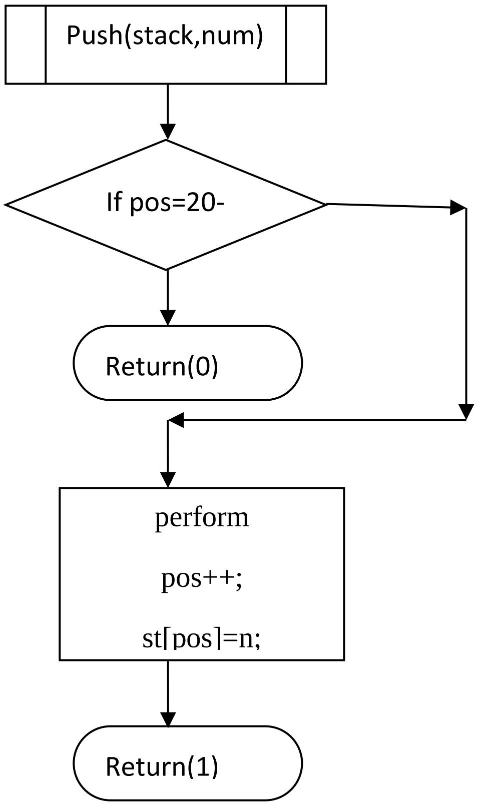

start
Initialization and declaration of variables and array
Display the message , .1push an element 2.pop an element 3.display stack element 4.exit

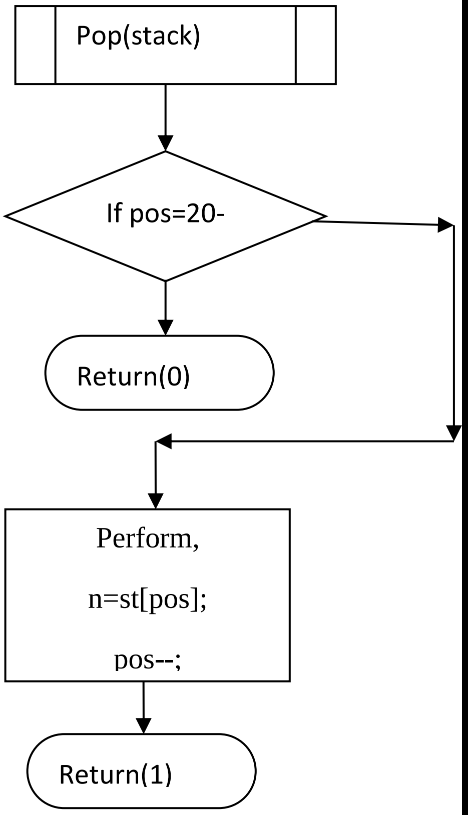

G
Read the value of choice of user
Switch(choice
F
If
If
If
C
Read the value of element to be inserted
Call,R=pop(stac)
If
(choice!=
Call,R=push(stack,num)
If (R==1)
E
D
Element is poped from the the stack
Stack is empty
If (R==1)
B
A
break


A
Element is pushed on the stack
break
C
Repeat for
Display the element of the stack
break
B
Print stack is full
break
D
Display you have entered wrong choice
break
E
F
Display do you want to continue and read the value of ch
Repeat while ch==1
G
stop

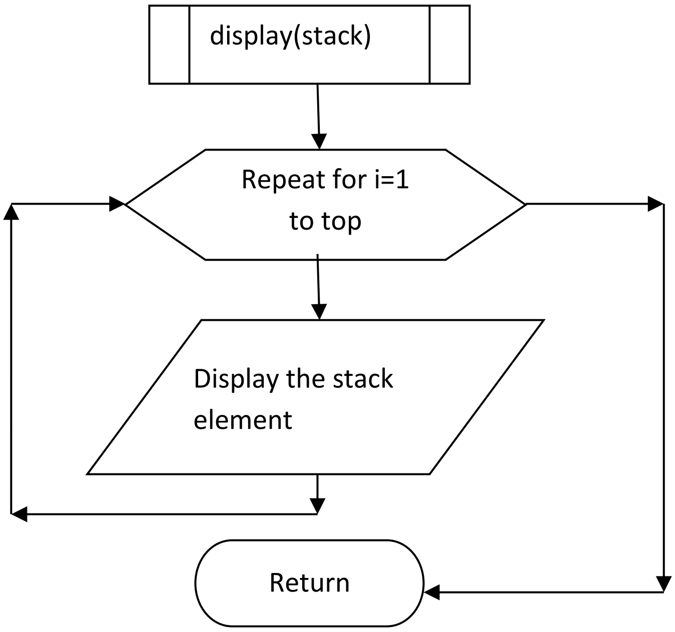
For push function

For pop function





For display function:



Program:

```
#include<conio.h>
#include<stdio.h>
int pos=-1;
void main()
{
    int stack[20],choice,num,r,ch;
    int push(int[],int);
    int pop(int[],int );
    void display(int[]);
    //void exit();
    clrscr();
    printf("\n1.push an element \n2.pop an element \n3.display stack element \n4.exit");
    do
    {
        printf("\nEnter your choice");
        scanf("%d",&choice);
        switch(choice)
        {
         case 1:
         printf("\nEnter any element");
         scanf("%d",&num);
         r=push(stack,num);
         if(r==1)
            printf("\nElement is pushed on the stack");
         else
            printf("\nstack is full");
         break;
         case 2:
         r=pop(stack);
         if(r==1)
```

```
          printf("\nElement is poped from the stack");
        else
        printf("\nstack is empty");
        break;
        case 3:
        display(stack);
        break;
        default:
        printf("you have entered wrong choice");
       }
       printf("\nDo you want to continue y/n ");
       scanf("%d",&ch);
       }
       while(ch==1);
       getch();
 }
 int push(int st[],int n )
 {


       if(pos==20-1)
       {
         return(0);
       }
       else
       {
         pos++;
         st[pos]=n;
         return(1);
         }
}
int pop(int st[])
```

```
   {
   if(pos==-1)
   {
    return(0);
   }
   else
   {
    st[pos]= -1;
    pos--;
    return(1);
  }
}
void display(int st[])
{
  int i;
  for(i=0;i<=pos;i++);
  {
     printf("%d\n",st[i]);
   }
 }
```

Output:

```
1 push an element
2 pop an element
3 display stack element
4 exit
 enter your choice1
enter any element10


 element pushed onthe stack
 do you want to continue y/n1
 enter your choice1
enter any element20

 element pushed onthe stack
 do you want to continue y/n1

 enter your choice1
enter any element30
 element pushed on  the stack
 do you want to continue y/n1

 enter your choice3
10
20
30

 do you want to continue y/n1

 enter your choice2

 enter is poped from the stack
 do you want to continue y/n1

 enter your choice3
10
20

 do you want to continue y/n 0
```