* Write a program to simulate the working of stack using an array with the following:
a) push
b) Pop
c) Display

the program should print appropiate messages for stack overflow, stack underflow

```
#include <stdio.h>
# include <stdio.h>
# define SIZE 5
void push (int);
void pop ();

int top = -1, input_array[SIZE];
void push ();
void pop ();
void Display ();

int main()
{
  int choice;
  while (1)
  {
    Printf ("\n Perform operations on the stack:");
    Prinf ("\n 1. push the Element \n2. pop the element
          \n3. Display \n4. end");
    Scanf ("%d", &choice);

    switch (choice)
    {
```

```
        case 1 :
            push ();
            break;
        Case 2 :
            pop ();
            break;
        Case 3 :
            Display ();
            break
        Case 4 :
            exit (0);
        default :
            printf ("\n Invalid choice !! ");
    }
}
}
void push ()
{
    int x ;
    if (top == SIZE - 1)
    {
    printf ("\n Overflow !! ");
    }
    Else
    {
    PHintf("\n Enter the element to be added onto the
            stack : ");
    Scanf ("%d", &x);
    top = top + 1;
    input_array [top] = x ;
    }
}
```

```
void pop()
{
 if (top == -1)
 {
  printf("\n Underflow !!");
 }
 Else
 {
  Printf("\n Popped element : %d", input_array[top]);
  top = top - 1;
 }
}
void Display()
{
  if (top == -1)
  {
    printf("\n Underflow!!");
  }
 Else
  {
    printf("\n Element present in the stack : \n");
    for(int i = top; i >= 0; --)
        printf("%d \n", input_array[i]);
  }
  return (0);
}
```

Output :

Perform operations on the stack :
1. Push the element
2. Pop the Element
3. Display
4. End.

Enter the choice : 1
Enter the element to be added onto the stack : 15

Enter the choice : 3
Elements present in the stack :
15

Enter the choice : 2
Poped Element : 15

Enter the choice : 3
Under flow !!

Enter the choice : 4
Exit.

1. Infix to postfix.

```
Voicl push (char symbol)
{
   top = top + 1
   stack [top] = symbol;
}

char Pop ()
{
 char symp;
 Symp = stack [top];
 top = top - 1
 return (symb);
}

int pred (char symbol)
{
 int p;
 Switch ( symbol )
 {
  Case 'n' : p = 3;
  break;
  Case '*';
  Case '%' : p = 2;
  break;
  Case '+';
  Case '-' : p-1;
    break;
  Case '(' : p = 0;
    break;
  Case '#' : P = -1
```

```
        break ;
}
return (p);
}

void infix to postfix ()
{
length = length (infix) ;
push ('#') ;
while (index I < length)
{
 Symbol = infix [index I];
 switch (symbol)
 {
  case '(' : Push (symbol);
   break
  case ')' : temp = Pop () ;
    while (temp != '(')
    {
    Postfix [pos] = temp ;
    pos ++;
    temp = pop () ;
    }
    break ;
Case '+' :
Case '-' :
Case '*' :
Case '/' :
Case '\n' ; while (Pred [stack [top]] >= pred (symbol)
    {
     temp = pop ();
     postfix [Post] = temp ;
    }
```

```
            Push (Symbol);
            break;
        default : Postfix [post++] = Symbol;
    }
    index++;
}
while (top>0)
{
    temp = pop();
    postfix [pos++] = temp;
}
}
```

Output.

Enter Infix Expression:

A+B*C+D

Infix expression:
A+B*C+D

Postfix expression:
ABC*+D+