```
enter the operation
 1.push
 2.pop
 3.display
 enter -1 to stop
1
enter the values
10
push operation is succesfull
1
enter the values
20
push operation is succesfull
2
20 pop() operation successfull
3
10
-1
stopping the operations

Process returned 0 (0x0)   execution time : 24.343 s
Press any key to continue.
```

DS LAB-2

PUSH POP AND DISPLAY OPERATOR

```
#include <stdio.h>
int stack [5], top = -1;
void main ()
{
Push ();
Pop ();
display ();
}

void Push()
{
int a;
Printf ("Enter the value of a\n");
scanf ("%d", &a);
top++;
Stack [top] = a;
}

void pop()
{
int temp;
temp = stack [top];
top --;
}
void display()
```

__/__/____

```
{
int i;
for (i = top; i>0; i--)
{
Printf ("%d\n", stack[top]);
}
}
```

output

1/1/2024

```
Enter size of stack 3
Assume the infix expression contains single letter variables and single digit constants only.

Enter Infix expression : k*l+m(n)
Postfix Expression: kl*mn+
Process returned 0 (0x0)   execution time : 49.048 s
Press any key to continue.
```

INFIX TO POSTFIX

__/__/____

```
#include <stdio.h>
#include <string.h>

int temp, index1 = 0, pos = 0, length;
char symbol, stack[20], infin[20],
postfin[20];
void push (char symbol);
char pop()
int preced (char symbol);

void main()
{
printf ("Enter the infin expression");
scanf ("%s", infin);
infin to postfin();
printf ("Enter the postfin expression");
}
void infin to postfin()
{
length = strlen (infin);
while (index1 <= length)
{
Symbol = infin [index1];
Switch (Symbol)
{
```

__/__/____

```
case'(' : push (symbol);
          break;
case')' : temp = pop();
          while (temp != '(')
          {
          postfix [pos] = temp;
          pos ++;
          temp = pop();
          }
          push (symbol);

case '+';
case '-';
case '*';
case '/';
case '^';
while (preced (stack [top]) >= preced
(symbol))
{
  temp = pop();
  postfix [pos] = temp;
    pos ++;
}
push (symbol);
default : postfix [pos++] = symbol;
}
```

```
inden 1++;
}
while (top >0)
{
temp= pop()
postfix[pos] = temp;
pos++;
}
}
void push (char Symbol)
{
 top ++;
stack[top] = Symbol;
}
Char pop()
{
 Char Symbol;
 Symbol = stack[top];
 top --;
return (Symbol);
}

nt preced (Char Symbol)

nt n;
```

__/__/____

```
switch (symbol)
{
 case '^' : n=3 ;
  break;
  case '*' :
  case '/' :    n = 2;
  case '+' :
  case '-' :    n=1;
  Case 'c' :   n =0;
}
return (n);
}
```