LAB - 1

1. Write a program to simulate the working of stack using an array with the following.

(a) Push

(b) POP

(c) Display

⇒
```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define MAX 3

int st[MAX], top=-1;
void push (int st[], int val);
int pop (int st[]);
void display (int st[]);

int main () {
    int val, option
    do {
        printf ("\n ***** MAIN MENU *****")
        printf ("\n 1. Push ");
        printf ("\n 2. POP ");
        printf ("\n 3. Display ");
        printf ("\n 4. Exit ");
        printf ("\n Enter your option : ");
        scanf (" %d", &option);
```

```c
switch (option)
{
    case 1 :
            printf ("\n Enter the number to be push
                on stack : ");
            scanf (" %d ", val);
            Push (st, val);
            break;
    case 2 :
        val = Pop (st);
            if (val != -1)
            printf ("\n The value deleted from stack
                is : %d ", val);
            break;
    case 3 :
            printf (" \n The stack is : ")
            display (st);
            break;
    }
}while (option != 4);
return 0;
}
```

```c
void push (int st[], int val)
{
    if (top == MAX-1)
    {
        printf(" \n stack overflow");
    }
    else
    {
        top++;
        st[top] = val;
    }
}

int POP (int st[])
{
    if (top == -1)
    {
        printf (" \n stack underflow");
        return -1;
    }
    else
    {
        val = st[top];
        top--;
        return val;
    }
}
```

```c
void display (int st[])
{
    int i ;
    if (top == -1)
        printf(" \n stack is empty");
    else
    {
        for(i=top; i>=0 ; i--)
        printf(" \n %d ", st[i]);
        printf(" \n") ;
    }
}
```

Q2 WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The Expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) & / (divide).

⟹

```c
# include <stdio.h>
# include <conio.h>
# include <ctype.h>
# include <string.h>
# define MAX 100


char st[MAX];
int top = -1;
void push (char st[], char val);
char pop (char st[]);
void Infixtopostfix (char source[], char target[]);
int getpriority (char);


int main () {
        char infix[100], postfix[100];
        clrscr();
        printf("\n Enter any infix Expression : ");
        scanf ("%s", infix[100]);
        strcpy (postfix, "");
        Infixtopostfix (infix, postfix);
```

```c
    printf ("\n The corresponding postfix expression is
    puts (postfix);
    getch ();
    return 0;
}
void Infixtopostfix (char source[], char target[])
{
    int i=0, j=0;
    char temp;
    strcpy (target, "");
    while (source[i] != '\0')
    {
        if (source[i] == '(')
        {
            push (st, source[i]);
            i++;
        }

        else if (source[i] == ')')
        {
            while ((top != -1) && (st[top] != '('))
            {
                target[j] = pop(st);
                j++;
            }
            if (top == -1)
            {
                pf ("\n Incorrect expression");
                exit (1);
```

```
        temp = pop(st) ;
        i++ ;
}
else if ( isdigit (source [i]) || isalpha (source [i]))
{
        target [j] = source[i] ;
        j++ ;
        i++ ;
}
else if (source [i] == '+' || source[i] == '-' || source[i]::*
    || source[i] == '/' || source[i] == '%')
{
        while ( (top! = -1) && (st[top] != '(') && (getpriority (
        st[top]) > getpriority (source [i])))
        {
                target [j] = pop(st) ;
                j++ ;
        }
        push (st, source[i]) ;
        i++ ;
}
else
{
        printf (" \n Incorrent element in expression ");
        exit(1);
}
```

```c
while ((top != -1) && (st[top] != '('))
{
        target[j] = POP(st);
        j++;
}
target[j] = '\0';
}


int getPriority (char op)
{
        if (op == '/' || op == '*' || op == '%')
                return 1;
        else if (op == '+' || op == '-')
                return 0;
}

void Push (char st[], char val)
{
        if (top == MAX -1)
                printf("\n stack overflow");
        else
        {       top++;
                st[top] = val;
        }
}
```

```
char pop (char st[])
{
    char val = '';
    if (top == -1)
        print ("\n stack underflow");
    else
    {
        val = st[top];
        top --;
    }
    return val;
}
```

01/01/2024

o/p : Enter any infix Expression : A+B
        The corresponding postfix expression is : AB+