VISHWAKARMA INSTITUTE OF TECHNOLOGY
DATA STRUCTURE ASSIGNMENT

NAME	Arpit Sudhir Vidhale
ROLL NO.	60
DIVISION	CS-D
BATCH	B3
PRN NO.	12111229

ASSIGNMENT 3

Question :

WAP to convert a given Infix expression into its equivalent Postfix expression and evaluate it using stack.

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
struct stack
{
    char data[15];
    int top;
};
int priority(char x)
{
    if (x == '(')
        return 0;
    if (x == '+' || x == '-')
        return 1;
    if (x == '*' || x == '/')
        return 2;
```

```
    return 0;
}
void push(struct stack *s, char x)
{
    s->top = s->top + 1;
    s->data[s->top] = x;
}
char pop(struct stack *s)
{
    char item;
    if (s->top == -1)
        return -1;
    else
    return(s->data[s->top--]);
}
int empty(struct stack *s)
{
    if (s->top == -1)
        return 1;
    return 0;
}
char top(struct stack *s)
{
    return s->data[s->top];
}
void init(struct stack *s)
{
    s->top = -1;
}
void post(char infix[], char
postfix[]) {
```

```
char x, y;
struct stack s;
init(&s);
int i, j = 0;
for (i = 0; infix[i] != '\0'; i++)
{
    x = infix[i];
    if (isalnum(x))
    {
        postfix[j++] = x;
    }
    else if (x == '(')
    {
        push(&s, x);
    }
    else if (x == ')')
    {
        while ((y = pop(&s)) != '(')
            postfix[j++] = pop(&s);
        printf("\n Postfix-j: %c",
    postfix[j]); }
    else
    {
        while (priority(x) <= priority(top(&s)))
            postfix[j++] = pop(&s);
        push(&s, x);
    }
}
while (!empty(&s))
    postfix[j++] = pop(&s);
postfix[j] = '\0';
```

```
}
int Eval(char *postfix)
{
    struct stack *s;
    int i = 0;
    int x1, x2, r = 0;


    for (i = 0; postfix[i] != '\0'; i++)
    {
        if (isalnum(postfix[i]))
        {
            push(s,postfix[i] - '0');
        }
        else
        {
            x2 = pop(s);
            x1 = pop(s);
            switch (postfix[i])
            {
            case '+':
                r = x1 + x2;
                break;
            case '-':
                r = x1 - x2;
                break;
            case '*':
                r = x1 * x2;
                break;
            case '/':
                r = x1 / x2;
                break;
```

```
            }
            push(s,r);
        }
    }
    return s->data[s->top];
}
int main()
{
    char inf[15], postf[15];
    printf("INFIX TO POSTFIX\n");
    printf("enter the infix expression : ");
    scanf("%s", inf);
    post(inf, postf);
    printf("postfix expression is %s \n", postf);
    printf("result is %d",Eval(postf));
}
```

Output :

```
arpit@arpit-HP:~/arpit$ cd "/home/arpit
INFIX TO POSTFIX
enter the infix expression : 5+4-3
postfix expression is 54+3-
result is 6
```

Stack implementation using array :

Code :

```
#include<stdio.h>
#define SIZE 6
int top = -1;
void push(int[]);
void pop(int[]);
```

```
int main() {
    int choice,x,stack[SIZE];


    do
    {
        printf("\n\t STACK OPERATIONS USING ARRAY");
        printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push(stack);
                break;
            }
            case 2:
            {
                pop(stack);
                break;
            }
            case 3:
            {
                display(stack);
                break;
            }
            case 4:
            {
                printf("\n\t EXIT POINT ");
                break;
            }
```

```
            default:
            {
                printf ("\n\t Please Enter a Valid
            Choice(1/2/3/4)"); }


        }
    }
    while(choice!=4);
    return 0;
}


void push(int stack[SIZE])
{
    int x;
    printf(" Enter a value to be pushed:");
    scanf("%d",&x);
    top++;
    stack[top]=x;
    return(1);
}


void pop(int stack[SIZE])
 {
    int n;
    if (top==-1)
    {
        printf("stack is empty");
        return(0);
    }
    else
    {
```

```
        printf("\n\t The popped elements is
        %d",stack[top]); n=stack[top];
        top--;
        return n;
    }
}

void display(int stack[SIZE])
{
    if(top>=0)
    {
        int i;
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }

}
```

Output :

```
        STACK OPERATIONS USING ARRAY
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:1
Enter a value to be pushed:5

        STACK OPERATIONS USING ARRAY
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:1
Enter a value to be pushed:4

        STACK OPERATIONS USING ARRAY
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:1
Enter a value to be pushed:3

        STACK OPERATIONS USING ARRAY
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:3

The elements in STACK

3
4
5
```

```
Press Next Choice
        STACK OPERATIONS USING ARRAY
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:2

        The popped elements is 3
        STACK OPERATIONS USING ARRAY
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:3

The elements in STACK

4
5
```

Stack implementation using Linkedlist :

<u>Code</u> :

```
#include <stdio.h>

#include <stdlib.h>

struct Node

{
```

```
    int data;
    struct Node *next;
} *top = NULL;
void push()
{
    int x;
    printf(" Enter a value to be pushed:");
    scanf("%d", &x);
    struct Node *t;
    t = (struct Node *)malloc(sizeof(struct Node));

    if (t == NULL)
        printf("stack is full\n");
    else
    {
        t->data = x;
        t->next = top;
        top = t;
    }
}
int pop()
{
    struct Node *t;
    int x = -1;

    if (top == NULL)
        printf("Stack is
    Empty\n"); else
    {
        t = top;
        top = top->next;
```

```
        x = t->data;
        free(t);
    }
    return x;
}
void Display()
{
    struct Node *p;
    p = top;
    while (p != NULL)
    {
        printf("%d ", p->data);
        p = p->next;
    }
    printf("\n");
}
int main()
{
      int choice, x, stack[10];
    do
    {
        printf("\n STACK OPERATIONS USING ARRAY");
        printf("\n 1.PUSH\t 2.POP\t
        3.DISPLAY\t4.EXIT"); printf("\n Enter the
        Choice:");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
        {
            push();
```

```
            break;
        }
        case 2:
        {
            pop();
            break;
        }
        case 3:
        {
            Display();
            break;
        }
        case 4:
        {
            printf("\n\t EXIT POINT ");
            break;
        }
        default:
        {
            printf("\n\t Please Enter a Valid Choice(1/2/3/4)");
        }
        }
    } while (choice != 4);
    return 0;
}
```

Output :

```
arpit@arpit-HP:~/arpit$ cd "/home/arpit/arpit/"

  STACK OPERATIONS USING ARRAY
  1.PUSH  2.POP   3.DISPLAY      4.EXIT
  Enter the Choice:1
  Enter a value to be pushed:45

  STACK OPERATIONS USING ARRAY
  1.PUSH  2.POP   3.DISPLAY      4.EXIT
  Enter the Choice:1
  Enter a value to be pushed:35

  STACK OPERATIONS USING ARRAY
  1.PUSH  2.POP   3.DISPLAY      4.EXIT
  Enter the Choice:1
  Enter a value to be pushed:25

  STACK OPERATIONS USING ARRAY
  1.PUSH  2.POP   3.DISPLAY      4.EXIT
  Enter the Choice:2

  STACK OPERATIONS USING ARRAY
  1.PUSH  2.POP   3.DISPLAY      4.EXIT
  Enter the Choice:3
 35 45

  STACK OPERATIONS USING ARRAY
  1.PUSH  2.POP   3.DISPLAY      4.EXIT
  Enter the Choice:4

        EXIT POINT arpit@arpit-HP:~/arpit$
```