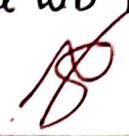


S.No	Component	Max. Marks	Marks Secured
1	Preparedness	2	2
2	Viva-Voce	2	2
3	Experiment	3	3
4	Analysis & Record	3	3
	Total	10	10
Date	10/6/22	Signature of the lab teacher 	

a) write a program to infix to postfix conversion.

Aim : To write a program to infix to postfix conversion

Program :

```

#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#define MAX50
struct stack
{
    int data [MAX];
    int top;
}
int percentage(char);
void init (struct stack *);
int empty (struct stack *);
int full (struct stack *);
int pop (struct stack *);
void push (struct stack *, int);
int top (struct stack *);
void infix - to - push (char infix [],
                        char postfix []);
void infix - to - postfix (char infix [],

```

```

    struct stacks;
    char x, taken;
    int i, j;
    init (&s);
    j = 0;
    for (i = 0; infix[i] != '\0'; i++)
    {
        taken = infix[i];
        if (isalnum(taken))
            postfix[j++] = taken;
        else
        {
            if (taken == '(')
                push (&s, '(');
            else
            {
                if (taken == ')')
                {
                    while ((x = pop(&s)) != '(')
                        postfix[j++] = x;
                    else
                }
            }
        }
    }

```

Register No : 464 Experiment No : 05 Date: 8/6/22

```

while (precedence(token) <= precedence
top(&s) && ! empty(&s))
{
    x = pop(&s);
    post fix [f++] = x;
}
push (&s) token);
}
while (! empty(&s))
{
    x = pop (&s)
    post fix [j++] = x;
}
post fix [j] = '\0';
}
int precedence (char x)
{
    int (x == '(')
    return(0);
    if (x == '+' || x == '-')
    return (1);
    if (x == '*' || x == '/')
    return(2);
    if (x == '^')
    return (3);
}
void init (struct stack *s)
{
    s->top = -1;
}
int empty (struct stack *s)
{
    if (s->top == -1)
    return (1);
    return (0);
}
void push (struct stack *s, int x)
{
    s->top = s->top + 1;
    s->data [s->top] = x;
}
int pop (struct stack *s)
{
    int x;
    x = s->data [s->top];
    s->top = s->top - 1;
    return(x);
}
int top (struct stack *s)
{
    return (s->data [s->top]);
}
void main ()
{
    char infx [30], postfix [30];
    printf("Enter an infix exp");
    gets(infix);
    infix -> postfix (infix, postfix);
    printf("In postfix exp: %s", postfix);
}

```


b) write a program to evaluate postfix expression ?

Aim: To write a program to evaluate postfix expression

Program:

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

```
#define MAX 20
```

```
struct Stack
```

```
{
    int data [MAX]
```

```
    int top;
```

```
};
```

```
int evaluate (char x, int op1,
              int op2)
```

```
{
```

```
    if (x == '+')
```

```
        return (op1 + op2);
```

```
    if (x == '-')
```

```
        return (op1 - op2);
```

```
    if (x == '*')
```

```
        return (op1 * op2);
```

```
    if (x == '/')
```

```
        return (op1 / op2);
```

```
    if (x == '%')
```

```
        return (op1 % op2);
```

```
}
```

```
void init (struct Stack *s)
```

```
{
```

```
    s->top = -1;
```

```
}
```

```
init (struct Stack *s)
```

```
{
```

```
    if (s->top == -1)
```

```
        return (1)
```

```
    else
```

```
        return (0);
```

```
}
```

```
int full (struct Stack *s)
```

```
{
```

```
    if (s->top == MAX-1)
```

```
        return (1);
```

```
    else
```

```
        return (0);
```

```
}
```

```
void push (struct Stack *s, int
```

```
{
```

```
    s->top = s->top + 1;
```

```
    s->data [s->top] = x;
```

```
}
```

```
int pop (struct Stack *s)
```

```
{
```

```
    int x;
```

```
    x = s->data [s->top];
```

```
    s->top = s->top - 1;
```

```
    return (x);
```

```
}
```

```
int main()
```

Register No :

646464

Experiment No :

05

Date:

8/6/22

```

{
    struct stack s;
    char x;
    int op1, op2, value;
    init(&s);
    printf("enter the expression in single digit operand &
           operation only");
    while(x = getch() != '\n');
    {
        if (is digit(x))
            push(&s, x - 48);
        else
        {
            op2 = pop(&s);
            op1 = pop(&s);
            val = evaluate(x, op1, op2);
            push(&s, val);
        }
    }
    value = pop(&s);
    printf("\n value of expression = %.d", value);
    return 0;
}

```

