LAB 2 PROGRAM

1. Write a program 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) and ^ (power).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 100

void push(char st[],char ch);
char pop(char st[]);
void infix_to_postfix(char src[],char ans[]);
int isalpha_numeric(char ch);
int isOperator(char ch);
int isPrior(char ch);
int top = -1;
char st[MAX];

int main(){
    char postfix[100],infix[100];
    printf("Enter the infix expression\n");
    scanf("%s",infix);
    strcpy(postfix,"");
    infix_to_postfix(infix,postfix);
    printf("The postfix expression is\n");
    printf("%s\n",postfix);
```

```
}
int isalpha_numeric(char ch){
    if((ch>= 'a' && ch<='z')||(ch >='A' && ch <= 'Z')||(ch >= '0' &&
ch <= '9')){
        return 1;
    }else{
        return 0;
    }
}
int isOperator(char ch){
    if(ch == '+' || ch == '-' || ch == '*' || ch == '/' ||ch == '%' ){
        return 1;
    }else{
        return 0;
    }
}
int isPrior(char ch){
    if( ch == '*' || ch == '/' ||ch == '%'){
        return 1;
    }else{
        return 0;
    }
}

void infix_to_postfix(char src[],char ans[]){
    int i=0;
    int j =0;
    while(src[i]!='\0') {
        if(src[i] == '('){
            push(st,src[i]);
        }
        else if(isalpha_numeric(src[i])){
            ans[j]= src[i];
            ++j;
        }
        else if(isOperator(src[i])){
            while(top != -1 && st[top] != '(' && (isPrior(st[top]) >=
isPrior(src[i]))){
                ans[j] = pop(st);
                ++j;
            }
            push(st,src[i]);
        }else if(src[i] == ')'){
            while(top != -1 && st[top] != '('){
                ans[j]= pop(st);
                ++j;
            }
            pop(st);
        }
        else{
            printf("invalid expression");
            exit(0);
        }
          ++i;

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

```
void push(char st[],char ch){
    if(top == MAX-1){
        printf("Stack overflow\n");
    }
    else{
        ++top;
        st[top] = ch;
    }
}
char pop(char st[]){
    char ch = '\0';
    if(top ==-1){
        printf("Stack underflow\n");
    }
    else{
        ch = st[top];
        --top;
    }
    return ch;
}
```

OUTPUT:

```
Enter the infix expression
(a+b/c*(d+e)-f)
The postfix expression is
abc/de+*+f-
```

2) WAP to simulate the working of a queue of integers using an array. Provide the following operations

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions.

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

#define MAX 5
int qu[MAX];
int front = -1;
int rear = -1;

void insert();
int delete_q();
void display();

int main(){
    while (1){
        int choice;
        printf("\n1. insert \t 2. delete \t 3. display \t 4. exit\n");
        scanf("%d", &choice);
        switch (choice){
        case 1:
            insert();
            break;
        case 2:
            delete_q();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        }
    }
}

void insert(){
    if (rear == MAX - 1){
        printf("Queue is Full\n");
        return;
    }
    printf("Enter the element to be inserted\n");
    int a;
    scanf("%d", &a);
    if (front == -1 && rear == -1){
        front = rear = 0;
    }
    else{
        rear++;
    }
    qu[rear] = a;
}

int delete_q(){
    if (front == -1){
        printf("Queue is Empty\n");
        return -1;
    }
    int x = qu[front];
    if (front == rear){
        front = rear = -1;
    }
    else{
        front++;
    }
    printf("The number popped is: %d\n", x);
    return x;
}

void display(){
    if (front == -1){
        printf("Queue is Empty\n");
        return;
    }
    printf("the elements are:\n");
    for (int i = front; i <= rear; i++){
        printf("%d \n", qu[i]);
    }
}
```

OUTPUT:

```
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
10
1. insert        2. delete       3. display      4. exit
```

```
1
Enter the element to be inserted
20
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
30
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
40

1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
50
1. insert        2. delete       3. display      4. exit
1
Queue is Full
1. insert        2. delete       3. display      4. exit
3
the elements are:
10
20
30
40
50
1. insert        2. delete       3. display      4. exit
2
The number popped is: 10
```

```
1. insert        2. delete       3. display      4. exit
2
The number popped is: 20
1. insert        2. delete       3. display      4. exit
2
The number popped is: 30
1. insert        2. delete       3. display      4. exit
2
The number popped is: 40
1. insert        2. delete       3. display      4. exit
2
The number popped is: 50
1. insert        2. delete       3. display      4. exit
2
Queue is Empty
1. insert        2. delete       3. display      4. exit
4
```

3) WAP to simulate the working of a circular queue of integers using an array.

Provide the following operations.

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions

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

#define MAX 5

int qu[MAX];
int front = -1;
int rear = -1;

void insert();
int delete_q();
void display();

int main(){
    while (1){
        int choice;
        printf("\n1. insert \t 2. delete \t 3. display \t 4. exit\n");
        scanf("%d", &choice);
        switch (choice){
        case 1:
            insert();
            break;
        case 2:
            delete_q();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        }
    }
}

void insert(){
    if ((front == 0 && rear == MAX - 1) || (front == rear + 1)){
        printf("Queue is Full\n");
        return;
    }
    printf("Enter the element to be inserted\n");
    int a;
    scanf("%d", &a);
    if (front == -1 && rear == -1){
        front = rear = 0;
    }
    else{
        rear = (rear + 1) % MAX;
    }
    qu[rear] = a;
}

int delete_q(){
    if (front == -1 && rear == -1){
        printf("Queue is Empty\n");
        return -1;
    }
    int x = qu[front];
    if (front == rear){
        front = rear = -1;
    }
    else{
        front = (front + 1) % MAX;
    }
    printf("The number poped is : %d\n", x);
    return x;
}

void display(){
    printf("the elements are:\n");
    if (front <= rear){
        for (int i = front; i <= rear; i++){
            printf("%d ", qu[i]);
        }
    }
    else{
        for (int i = front; i < MAX; i++){
            printf("%d ", qu[i]);
        }
        for (int i = 0; i <= rear; i++){
            printf("%d ", qu[i]);
        }
    }
    printf("\n");
}
```

OUTPUT :

```
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
2
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
4
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
6
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
8
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
18
1. insert        2. delete       3. display      4. exit
1
Queue is Full
```

```
1. insert        2. delete       3. display      4. exit
3
the elements are:
2 4 6 8 18

1. insert        2. delete       3. display      4. exit
2
The number poped is : 2
1. insert        2. delete       3. display      4. exit
1
Enter the element to be inserted
100
1. insert        2. delete       3. display      4. exit
3
the elements are:
4 6 8 18 100
1. insert        2. delete       3. display      4. exit
4
```