LAB3

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 <ctype.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100
char st[MAX];
int top = -1;
void push(char st[], char);
char pop(char st[]);
void InfixtoPostfix(char source[], char target[]);
int getpri(char);

void main()
{
   char infix[100], postfix[100];
   printf("\n Enter any infix expression : ");
   gets(infix);
   strcpy(postfix, "");
   InfixtoPostfix(infix, postfix);
   printf("\n The corresponding postfix expression is : ");
   puts(postfix);
}

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)
        {
            printf("\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] == '%' || source[i] == '^')
    {
        while ((top != -1) && (st[top] != '(') && (getpri(st[top]) > getpri(source[i])))
        {
            target[j] = pop(st);
            j++;
        }
        push(st, source[i]);
        i++;
    }
    else
    {
        printf("\n INCORRECT ELEMENT IN EXPRESSION");
        exit(1);
    }
}
while ((top != -1) && (st[top] != '('))
{
    target[j] = pop(st);
    j++;
}
target[j] = '\0';
```

```
}
int getpri(char op)
{
    if (op == '^')
        return 2;
    else 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)
        printf("\n STACK UNDERFLOW");
    else
    {
        val = st[top];
        top--;
    }
    return val;
}
```

Output:

Output Clear

```
/tmp/1KOKpxfvK5.o
Enter any infix expression : A+B*C/D-F+A^E
The corresponding postfix expression is : ABCD/*FAE^+-+
```

Program-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 N 4
int q[N];
int REAR = -1;
int FRONT = -1;
void enq();
void deq();
void display();
void enq() {
   if (REAR == N - 1) {
      printf("Overflow!\n");
   } else {
      int item;
      printf("Enter the element to insert:\n");
      scanf("%d", &item);
      if (REAR == -1 && FRONT == -1) {
         REAR++;
         q[REAR]=item;
         FRONT++;
      }
      else{
      REAR++;
      q[REAR] = item;
      }
   }
}
void deq() {
   int val;
   if (FRONT == -1 || FRONT > REAR) {
      printf("Queue empty!\n");
   } else {
      val = q[FRONT];
      FRONT++;
      printf("Element deleted is %d\n", val);
   }
}
void display() {
   int i;
   for (i = REAR; i >= FRONT; i--) {
      printf("%d\n", q[i]);
```

```
    }
}
int main() {
    int choice;
    while (1) {
        printf("Enter 1 to add, 2 to delete, 3 to display queue, any other key to exit:\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                enq();
                break;
            case 2:
                deq();
                break;
            case 3:
                display();
                break;
            default:
                printf("Invalid key entered\n");
                exit(1);
        }
    }
    return 0;
}
```

Output

```
/tmp/1KOKpxfvK5.o
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
1
Enter the element to insert:
1
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
1
Enter the element to insert:
2
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
1
Enter the element to insert:
3
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
1
Enter the element to insert:
4
```

Output

```
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
1
Overflow!
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
3
4
3
2
1
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
2
Element deleted is 1
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
2
Element deleted is 2
```

```
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
2
Element deleted is 3
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
2
Element deleted is 4
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
2
Queue empty!
Enter 1 to add
2 to delete
3 to display queue
any other key to exit:
4
Invalid key entered
```

Program 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 N 3
```

```
int q[N];
int REAR=-1;
int FRONT=-1;
void enq();
void deq();
void display();

void enq(){
   int item;
   printf("enter element to insert:\n");
   scanf("%d",&item);
   if(FRONT==-1 && REAR==-1){
      FRONT=REAR=0;
      q[REAR]=item;
   }
   if((REAR+1)%N==FRONT){
      printf("queue overflow!\n");
   }
   else{
      REAR=(REAR+1)%N;
      q[REAR]=item;
   }
}
void deq(){
   if(FRONT==-1 && REAR==-1){
      printf("empty queue!\n");
   }
   else if(FRONT==REAR){
      printf("the deleted element is: %d\n",q[FRONT]);
      FRONT=REAR=-1;
   }
   else{
      printf("deleted element:%d\n",q[FRONT]);
      FRONT=(FRONT+1)%N;
   }
}
void display(){
   int i;
   for(i=FRONT;i<=REAR;i++){
      printf("%d\t",q[i]);
   }
}
void main(){
   int choice;
```

```
    while(1){
    printf("enter 1. insert 2. delete 3. display\n");
    scanf("%d",&choice);
    switch(choice){
        case 1: enq();
            break;
        case 2: deq();
            break;
        case 3: display();
            break;
        default: printf("invalid entry\n");
            exit(0);
    }
    }
}
```

Output

Output

```
/tmp/lrehIOhobV.o
enter 1. insert 2. delete 3. display
1
enter element to insert:
1
enter 1. insert 2. delete 3. display
1
enter element to insert:
2
enter 1. insert 2. delete 3. display
1
enter element to insert:
3
queue overflow!
enter 1. insert 2. delete 3. display
2
deleted element:1
enter 1. insert 2. delete 3. display
1
enter element to insert:
3
enter 1. insert 2. delete 3. display
3
```

```
1 2 3
```