Infix to Postfix (LAB-3)

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

#define MAX 100
char stack[MAX];
int top = -1;

void push(char);
char pop();
int precedence(char);
void infixtopostfix(char infix[], char postfix[]);

void push(char x)
{
    if (top == MAX-1)
    {
        printf("overflow!\n");
        exit(EXIT_FAILURE);
    }
    else
    {
        top++;
        stack[top] = x;
    }
}
char pop()
{
    if (top == -1)
    {
        printf("underflow\n");
    }
    else
    {
        char popped = stack[top];
        top--;
```

```
    return popped;
  }
}
int precedence (char symbol)
{
  if (symbol == '^')
      return 3;
  else if (symbol == '*' || symbol == '/')
      return 2;
  else if (symbol == '+' || symbol == '-')
      return 1;
  else
      return -1;
}
void infix to postfix (char infix[], char postfix[])
{
  int i=0; j=0;
  char symbol, temp;
  push ('#');
  while ((symbol = infix[i++]) != '\0')
  {
    if (symbol == '(')
        push (symbol);
    else if (isalnum (symbol))
        postfix [j++] = symbol;
    else if (symbol == ')')
    {
      while (stack [top] != '(')
      {
        postfix [j++] = pop();
      }
      temp = pop();
    }
```

```
else
{
    while(precedence(stack[top]) >= precedence(symbol))
    {
        postfix[j++] = pop();
    }
    push(symbol);
}
}
while(stack[top] != '#')
{
    postfix[j++] = pop();
}
postfix[j] = '\0';
}

int main()
{
    char infix[MAX], postfix[MAX];
    printf("Enter a valid parenthesized infix exp:\n");
    scanf("%s", infix);
    infix to postfix(infix, postfix);
    printf("The postfix exp is: %s\n", postfix);
    return 0;
}
```

Output:

Enter a valid parenthesized infix exp:

a*b+c*d-e

The postfix exp is: ab*cd*+e-

ALP

28/12/2023

Postfix evaluation (LAB-3)

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define max 20
int stack[20];
int top=-1;
void push (int a)
{
  stack[++top]=a;
}
int pop()
{
  return stack[top--];
}
void main()
{
  char postfix[max];
  printf("Enter postfix expression: ");
  scanf("%s", postfix);
  int res=0, a, b;
  for(int i=0; i<strlen(postfix); i++)
  {
    if(isalnum(postfix[i]))
        push(postfix[i]-'0');
    else
    {
        b=pop();
        a=pop();
```

```
switch (postfix[i])
{
 case '+': push(a+b);
           break;
 case '-': push(a-b);
           break;
 case '*': push(a*b);
           break;
 case '/': push(a/b);
           break;
 case '^': push(a^b);
           break;
    }
  }
  }
  res = pop()
  printf("%s = %d", postfix, res);
}
```

Output:

Enter postfix expression : 23*31*+5-

23*31*+5- = 4

## Linear Queue (LAB-3)

```
#include<stdio.h>
#define size 30

int queue[size];
int front = -1;
int rear = -1;

void insert(int a)
{
  if (rear == size-1)
  {
    printf("Queue overflow\n");
    return;
  }
  else
  {
    if (front == -1)
        front = 0;
    queue[++rear] = a;
  }
}

void delete()
{
  if (front == -1 || front > rear)
  {
    printf("Queue Empty\n");
    ;
  }
  printf("Queue:");
  for(int i = front; i <= rear; i++)
  {
    printf("%d", queue[i]);
  }
  else { front++;
  }
}

void display() {
if (front == -1) {
    printf("Queue Empty");
    return;
}
printf("Queue:");
for(int i=front; i<=rear;
    i++)
  {
    printf("%d",queue[i]);
  }
}
```

```
void main()
{
  int choice;
  int a;
  while(1)
  {
    printf("\n1. Insert \n2. Delete \n3. Display \n Choice:");
    scanf("%d", &choice);

    switch(choice)
    {
      case 1: printf("Enter an element:");
              scanf("%d", &a);
              insert(a);
              display();
              break;

      case 2: delete();
              display();
              break;

      case 3: display();
              break;
    }
  }
}
```

Output

1. Insert
2. Delete
3. Display

Choice : 1

Enter an element : 7

Queue : 7

1. Insert
2. Delete
3. Display
Choice: 1
Enter an element: 8
Queue: 7 8

1. Insert
2. Delete
3. Display
Choice: 2
Queue: 8

1. Insert
2. Delete
3. Display
Choice: 3
Queue: 8

## Circular Queue (LAB-3)

```
#include <stdio.h>
#define size 5
int queue[size];
int front = -1;
int rear = -1;

void enqueue (int a)
{
  if ((front == rear + 1) || (front == 0 && rear == size - 1))
  {
    printf ("Queue overflow \n");
    return;
  }
```