Lab-3

28/12/23

1) // Program to convert infix to postfix

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

char stack[size];
int top = -1;

void push(char a){
    stack[++top] = a;
}

char pop(){
    return stack[top--];
}

int precedence(char a){
    if(a=='^')
        return 3;
    else if(a=='*' || a=='/')
        return 2;
    else if(a=='+' || a=='-')
        return 1;
    else
        return 0;
}
```

```
void main(){
char infix[size], postfix[size];
int j=0;
for(int i=0; i<strlen(infix); i++){
        if(infix[i]=='(')
            push('(');
        else{
            if(precedence[stack[top]) < precedence[infix[i]]
                push(infix[i]);
            else{
                while(stack[top]!='('){
                    postfix[j++] = pop();
                }
                push[infix[i]];
            }
        }
}
while(top!=-1){
    postfix[j++] = pop();
}
postfix[i]='\0';
printf("Postfix expression is: %s", postfix);
```

Output:

Enter Expression: A*B+C*D-E

Postfix Expression is AB*CD*+E-

2) // Evaluating Postfix Expression

```
#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]="12*34*+5-";
    int result=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;
        }
    }
}
result = pop();
printf("%s = %d", postfix, result);
}
```

Output:

Enter postfix expression: 12*34*+5-

12*34*+5- = 9

5) // Program to implement Queue

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

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

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

void delete(){
    if(front==-1 || front>rear){
        printf("Queue Empty\n");
    else
        front++;
}

void display(){
    if(front==-1)
        printf("Queue Empty\n");
        return;
    printf("Queue:");
```

```
        for(int i = front; i<=rear; i++){
            printf("%d ", queue[i]);
        }
        printf("\n------\n");
}

void main(){
    int choice, a;
    while(1){
        printf("Queue Operations\n Insert\n Delete\n Display");
        scanf("%d", &choice);
        switch(choice)
        {
          case 1;
              scanf("%d", &a);
              insert(a);
              display();
              break;

          case 2;
              delete();
              display();
              break;
          case 3;
              display();
              break;

    }
}
```

Output:

Queue Operations
1. Insert
2. Delete
3. Display
choice: 1
Enter Element: 3
Queue: 3
choice: 1
Enter Element: 4
Queue: 3 4
choice: 2
Queue: 4
choice: 3
Queue: 4

28/12/23