LAB 2 PROGRAMS

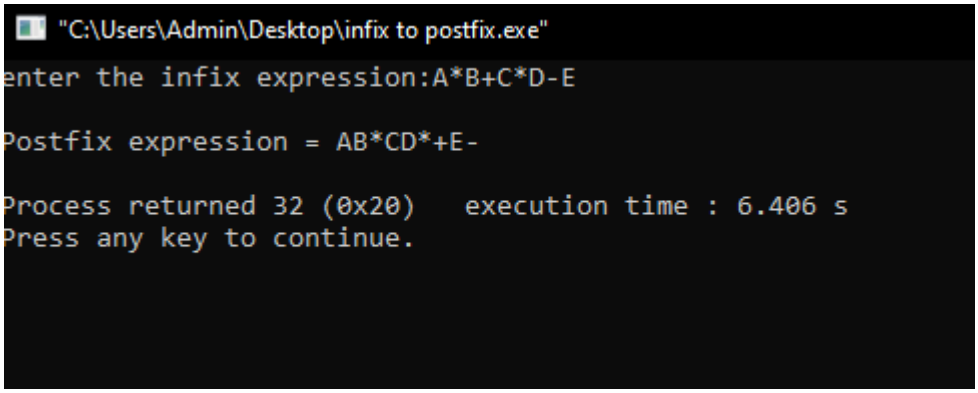
PROGRAM-1 : INFIX TO POSTFIX EXPRESSION

SOURCE CODE:

```
#include <stdio.h>
#include<ctype.h>
#define SIZE 50
char stack[SIZE];
int top=-1;
push(char ele)
{
   stack[++top]=ele;
}
char pop()
{
   return(stack[top--]);
}
int pr(char symbol)
{
   if (symbol == '^')
   {
      return(3);
   }
   else if (symbol == '*' || symbol == '/')
   {
      return(2);
   }
   else if (symbol == '+' || symbol == '-')
   {
      return(1);
   }
   else
   {
      return (0);
   }
}
void main()
{
   char infix[50],postfix[50],ch,ele;
   int i=0,k=0;
   printf("enter the infix expression:");
   scanf("%s",infix);
   push('#');
   while( (ch=infix[i++]) != '\0')
   {
      if(ch=='(') push(ch);
      else
```

```
        if(isalnum(ch)) postfix[k++]=ch;
        else
          if(ch ==')')
          {
             while(stack[top]!='(')
                postfix[k++]=pop();
             ele=pop();
          }
          else
          {
             while(pr(stack[top])>=pr(ch))
              postfix[k++]=pop();
             push(ch);
          }
   }
   while(stack[top]!='#')
      postfix[k++]=pop();
   postfix[k]='\0';
   printf("\nPostfix expression = %s\n",postfix);
}
```

OUTPUT:



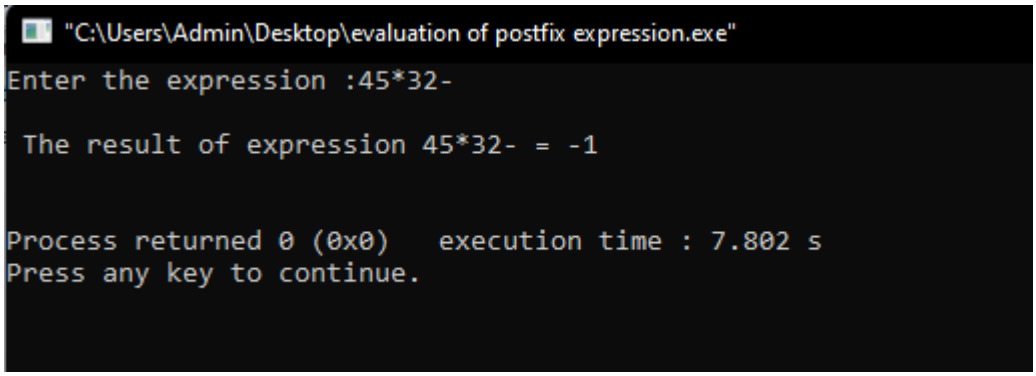
PROGRAM-2 : EVALUATION OF POSTFIX EXPRESSION

SOURCE CODE:

```
#include<stdio.h>
int stack[20];
int top = -1;
 void push(int x)
 {
  stack[++top]=x;
  }
  int pop()
  {
  return stack[top--];
  }
  int main()
  {
  char exp[20];
  char *e;
  int n1,n2,n3,num;
  printf("Enter the expression :");
  scanf("%s",exp);
  e=exp;
  while(*e!='\0')
  {
  if(isdigit(*e))
  {
  num=*e-48;
  push(num);
  }
  else
  {
  n1=pop();
  n2=pop();
  switch(*e)
  {
  case '+':
  {
  n3=n1+n2;
  break;
  }
  case '-':
  {
  n3=n1-n2;
  break;
  }
  case '*':
  {
  n3=n1*n2;
```

```
break;
}
case '/':
{
n3=n1/n2;
break;
}
}
push(n3);
}
e++;
}
printf("\n The result of expression %s = %d\n\n",exp,pop());
}
```

OUTPUT:



LAB 3 PROGRAMS

PROGRAM-1 : QUEUE

SOURCE CODE:

```
#include<stdio.h>
#define MAX 50
int queue_array[MAX];
int rear=-1;
int front=-1;
display()
{
  int i;
  if(front==-1)
    printf("queue is empty\n");
else
{
  printf("queue is :\n");
  for(i=front;i<=rear;i++)
    printf("%d",queue_array[i]);
  printf("\n");
}
}
main()
{
  int choice;
  while(1)
  {
    printf("1.insert\n");
    printf("2.delete\n");
    printf("3.display\n");
    printf("4.exit\n");
    printf("enter your choice:");
    scanf("%d",&choice);
    switch(choice)
    {
      case 1:
      insert();
      break;
      case 2:
      delete();
      break;
      case 3:
      display();
      break;
      case 4:
      exit(1);
      break;
```

```
      default:
      printf("invalid choice\n");
    }
  }
}
insert()
{
  int add_item;
  if(rear==MAX-1)
    printf("queue overflow\n");
  else
  {
    if(front==-1)
    front=0;
    printf("insert the element in the queue:");
    scanf("%d",&add_item);
    rear+=1;
    queue_array[rear]=add_item;
   }
}
delete()
{
  if(front==-1 || front>rear)
  {
    printf("queue underflow\n");
    return;
  }
  else
  {
    printf("deleted element is : %d\n",queue_array[front]);
    front+=1;
  }
}
```

OUTPUT:

```
C:\Users\Admin\Desktop\QUEUE.exe
1.insert
2.delete
3.display
4.exit
enter your choice:1
insert the element in the queue:19
1.insert
2.delete
3.display
4.exit
enter your choice:1
insert the element in the queue:28
1.insert
2.delete
3.display
4.exit
enter your choice:3
queue is :
1928
1.insert
2.delete
3.display
4.exit
enter your choice:2
deleted element is : 19
1.insert
2.delete
3.display
4.exit
enter your choice:3
queue is :
28
1.insert
2.delete
3.display
4.exit
enter your choice:4

Process returned 1 (0x1)   execution time : 219.333 s
Press any key to continue.
```

PROGRAM 2: CIRCULAR QUEUE

SOURCE CODE:

```
#include<stdio.h>

#define SIZE 5

int items[SIZE];
int front = -1, rear = -1;

int isFull() {
   if ((front == rear + 1) || (front == 0 && rear == SIZE - 1))
      return 1;
   return 0;
}

int isEmpty() {
   if (front == -1)
      return 1;
   return 0;
}

void enQueue(int element) {
   if (isFull())
      printf("\nQueue is full");
   else {
      if (front == -1)
         front = 0;
      rear = (rear + 1) % SIZE;
      items[rear] = element;
      printf("\nInserted -> %d", element);
   }
}

int deQueue() {
   int element;
   if (isEmpty()) {
      printf("\nQueue is empty");
      return -1;
   } else {
      element = items[front];
      if (front == rear) {
         front = -1;
         rear = -1;
      } else {
         front = (front + 1) % SIZE;
      }
      printf("\nDeleted element -> %d\n", element);
```

```
        return element;
    }
}

void display() {
    int i;
    if (isEmpty())
        printf("\nEmpty queue\n");
    else {
        printf("\nFront -> %d", front);
        printf("\nItems -> ");
        for (i = front; i != rear; i = (i + 1) % SIZE) {
            printf("%d ", items[i]);
        }
        printf("%d", items[i]);
        printf("\nRear -> %d\n", rear);
    }
}

int main() {
    enQueue(1);
    enQueue(2);
    enQueue(3);
    enQueue(4);
    enQueue(5);

    display();

    deQueue();
    deQueue();

    display();

    enQueue(6);
    enQueue(7);

    display();

    return 0;
}
```

<u>OUTPUT:</u>

```
C:\Users\Admin\Desktop\CQ.exe

Inserted -> 1
Inserted -> 2
Inserted -> 3
Inserted -> 4
Inserted -> 5
Front -> 0
Items -> 1 2 3 4 5
Rear -> 4

Deleted element -> 1

Deleted element -> 2

Front -> 2
Items -> 3 4 5
Rear -> 4

Inserted -> 6
Inserted -> 7
Front -> 2
Items -> 3 4 5 6 7
Rear -> 1

Process returned 0 (0x0)   execution time : 0.002 s
Press any key to continue.
```