Experiment No 2.1

Date:

Stack implementation using array

AIM: Implement a stack using an array.

ALGORITHM

PROGRAM

```
#include<stdio.h>
#define MAX 5
int stack[MAX],top=-1;
void push(int data)
{
   if(top==MAX-1)
      printf("Stack Overflow\n");
   else
   stack[++top]=data;
}
void pop()
{
   int d;
   if(top==-1)
      printf("Stack Empty\n");
   else
   {
      d=stack[top--];
      printf("Poped element : %d\n",d);
   }
}
void display()
{
   if(top==-1)
      printf("Stack Empty\n");
   else
   {
      printf("\n");
      for(int i=top;i>=0;i--)
      {
         printf("%d\n",stack[i]);
      }
   }
}
int main()
{
   int ch,element;
   do
   {
      printf("\nChoose operation");
      printf("\n1.Push");
      printf("\n2.Pop");
      printf("\n3.Display Stack");
      printf("\n4.Exit");
      printf("\nEnter an option\n");
      scanf("%d",&ch);
```

```
        switch(ch)
        {
            case 1:
                printf("\nEnter the element to insert\n");
                scanf("%d",&element);
                push(element);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
        }
    }while(ch<4);
    return 0;
}
```

OUTPUT

```
Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
1

Enter the element to insert
3

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
1

Enter the element to insert
4

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
1
```

```
Enter the element to insert
5

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
1

Enter the element to insert
6

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
1

Enter the element to insert
7

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
1

Enter the element to insert
9
Stack Overflow

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
3

7
6
```

```
5
4
3

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
2
Poped element : 7

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
2
Poped element : 6

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
2
Poped element : 5

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
2
Poped element : 4

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
2
Poped element : 3
```

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
2
Stack Empty

Choose operation
1.Push
2.Pop
3.Display Stack
4.Exit
Enter an option
4

Experiment No 2.2

Date:

Infix to postfix conversion

AIM: Program to convert the Infix expression to postfix expression

ALGORITHM

PROGRAM

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100
char stack[MAX],postfix[MAX],infix[MAX];
int top = -1;

void push(char x)
{
     if(top==MAX-1)
        printf("Stack Overflow\n");
    else
    stack[++top] = x;
}

char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char symbol)
{
    switch(symbol)
    {
        case '+':
            return 1;
        case '-':
            return 1;

        case '*':

        case '/':
            return 2;
        case '^':
           return 3;
        default:
            return 0;

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

}

void intopost(){
    int i,j=0;
    char symbol,next;
    for(i=0;i<strlen(infix);i++){
        symbol=infix[i];
        // if(space(symbol)==1)
        {
```

```
        switch(symbol){
            case '(':
               push(symbol);
               break;
            case ')':
               while((next=pop())!='('
                  postfix[j++]=next;
                  break;
            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
               while(!isEmpty() && priority(stack[top])>=priority(symbol))
                   postfix[j++]=pop();
               push(symbol);
               break;
            default:
                postfix[j++]=symbol;
        }

        }
    }
    while(!isEmpty())
    {
        postfix[j++]=pop();
    }
    postfix[j]='\0';
}
    void postprint()
    {
        int i;
        printf("\nthe equivalent postfix infixression is ");

        for(i=0;postfix[i]!='\0';i++)
        {
            printf("%c",postfix[i]);
        }
        printf("\n");

    }


int main()
{
    printf("Enter the infix expression : ");
    gets(infix);
    intopost();
    postprint();


    return 0;
}
```

OUTPUT

Enter the infix expression : (A+(B*C-(D/E-F)*G)*H)
the equivalent postfix infixression is ABC*DE/F-G*-H*

Experiment No 2.3

Date:

Postfix Evaluation

AIM: Program to evaluate the postfix expression entered by the user

ALGORITHM

PROGRAM

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
int stack[20];
int top = -1;

void push(int x)
{
   stack[++top] = x;
}

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

int main()
{
   char exp[100];
   char *e;
   int n1,n2,n3,num,ch;
   printf("Enter the postfix 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 = n2 - n1;
            break;
         }
         case '*':
         {
            n3 = n1 * n2;
            break;
         }
         case '/':
         {
            n3 = n2 / n1;
            break;
         }
         }
         push(n3);
```

```
        }
        e++;
    }
    printf("\nThe result of expression  =  %d\n\n",pop());
    return 0;
}
```

OUTPUT

```
Enter the postfix expression :: 623+*5/2+
The result of expression  =  8
```

Experiment No 2.4

Date:

Queue implementation using Array

AIM: Implement a queue using an array.

ALGORITHM

PROGRAM

```
#include <stdio.h>
#include<stdlib.h>
#define MAX 4
int qarray[MAX];
int front=-1,rear=-1;

int isEmpty(){
    if(front ==rear+1||front==-1){
        return 1;
    }
    else{
        return 0;
    }
}
int isFull(){
    if (rear== MAX-1)
       return 1;
    else
       return 0;
}
void enqueue(int data){
     int t;
    t=isFull();
    if(t==1){
        printf("Cannot add the queue is full");

    }
    if(t==0){
    if (front==-1)
        front=0;

        qarray[++rear]=data;
    }

}
int dequeue(){
    int t;
    t=isEmpty();
    if(t==1){
        printf("Queue is empty");
        return -99;
    }
    if(t==0){
        int data;
        data=qarray[front ];
        front++;
       return data;
    }

}

int peek(){
    int t;
    t=isEmpty();
    if(t==1){
        printf("Queue is empty");
        return -99;
```

```
    }return qarray[front];

}

void display(){
    int t;
    t=isEmpty();
    if(t==1){
        printf("Queue is empty");
    }
    if(t==0){
         printf("Queue is");
        printf("\n");
        for(int i=front;i<=rear;i++)
        {
            printf("%d\n",qarray[i]);
        }
    }

}

int main(){
    int ch,element,top,temp;
    do{
        printf("\nChoose operation");
        printf("\n1.Enqueue");
        printf("\n2.Dequeue");
        printf("\n3.Display Queue");
        printf("\n4.Peek");
        printf("\n5.Exit");
        printf("\nEnter an option\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
             printf("\nEnter the element to insert\n");
                scanf("%d",&element);
                enqueue(element);
                break;
            case 2:

            temp=dequeue();
            if(temp!=-99)
            printf("The dequeued element is %d",temp);
            break;
            case 3:
                display();
                break;
            case 4:

               top=peek();
               if(top!=-99)
               printf("The element at the top is %d",top);
               break;

        }
    }while(ch<5);
     return 0;
}
```

OUTPUT

```
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
1

Enter the element to insert
2

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
1

Enter the element to insert
4

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
1

Enter the element to insert
3

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
1

Enter the element to insert
6
```

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
1

Enter the element to insert
7
Cannot add the queue is full
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
4
The element at the top is 2
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
3
Queue is
2
4
3
6

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
2
The dequeued element is 2
Choose operation
1.Enqueue
2.Dequeue

3.Display Queue
4.Peek
5.Exit
Enter an option
2
The dequeued element is 4
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
2
The dequeued element is 3
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
2
The dequeued element is 6
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option
2
Queue is empty
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Peek
5.Exit
Enter an option

Experiment No 2.5

Date:

CIRCULAR QUEUE

AIM:Implement a circular queue using array

ALGORITHM

PROGRAM

```
#include <stdio.h>
#include<stdlib.h>
#define MAX 4
int qarray[MAX];
int front=-1,rear=-1;

int isEmpty(){
    if(front == -1){
        return 1;
    }
    else{
        return 0;
    }
}
int isFull(){
    if( (front == 0 && rear == MAX - 1) || (front == rear+1))
      return 1;
    else
      return 0;
}
void enqueue(int data){
     int t;
    t=isFull();
    if(t==1){
        printf("Cannot add the queue is full");

    }
    if(t==0){
    if (front == -1)
        front=0;
    if(rear == MAX-1)
      rear=0;
    else
     rear=rear+1;

    qarray[rear]=data;

    }
}
int dequeue(){
    int t,data;
    t=isEmpty();
    if(t==1){
        printf("Queue is empty");
        return -99;

    }

    if(t==0){
    data=qarray[front];
    if(front==rear){
      front=rear=-1;
       }
    else if (front==MAX-1){
           front=0;
       }
    else
        front=front+1;
    return data;
```

```
    }


}

void display(){
    int t,temp;
    t=isEmpty();
    if(t==1){
        printf("Queue is empty");



    }
    if(t==0){
    temp=front;
    if(front<=rear){
      while(temp<=rear){
        printf("%d\t",qarray[temp]);
        temp++;
      }
    }
    else
    {
        while(temp<=MAX-1){
             printf("%d\t",qarray[temp]);
            temp++;
        }
        temp=0;
        while(temp<=rear){
             printf("%d\t",qarray[temp]);
            temp++;
        }

    }
    }
  printf("\n");

}

int main(){
    int ch,element,top,temp;
    do{
        printf("\nChoose operation");
        printf("\n1.Enqueue");
        printf("\n2.Dequeue");
        printf("\n3.Display Queue");
        // printf("\n4.Peek");
        printf("\n4.Exit");
        printf("\nEnter an option\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
             printf("\nEnter the element to insert\n");
                scanf("%d",&element);
                enqueue(element);
                break;
            case 2:
```

```
            temp=dequeue();
            if(temp!=-99)
            printf("The dequeued element is %d",temp);
            break;
            case 3:
               display();
               break;

        }
    }while(ch<4);
     return 0;
}
```

OUTPUT

```
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
1

Enter the element to insert
2

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
1

Enter the element to insert
5

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
1

Enter the element to insert
3

Choose operation
1.Enqueue
2.Dequeue
```

```
3.Display Queue
4.Exit
Enter an option
1

Enter the element to insert
8

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
1

Enter the element to insert
6
Cannot add the queue is full
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
3
2       5       3       8

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
2
The dequeued element is 2
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
2
The dequeued element is 5
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
```

4.Exit
Enter an option
2
The dequeued element is 3
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
2
The dequeued element is 8
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
2
Queue is empty
Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
3
Queue is empty

Choose operation
1.Enqueue
2.Dequeue
3.Display Queue
4.Exit
Enter an option
4