**Assignment 6 | 9th January 2021**

Name: Rajarshi Lodh

Email: [rajarshilodh@gmail.com](mailto:rajarshilodh@gmail.com)

**Question 1**

Write a program implementing insert, delete and display operation of Circular Queue.

Program code:

#include<stdio.h>

#define MaxSize 5

struct queue

{

    int arr[MaxSize];

    int rear,front;

};

void enqueue(struct queue \*q);

int dequeue(struct queue \*q);

void display(struct queue q);

int main()

{

    struct queue q;

    q.front=-1;

    q.rear=-1;

    int o,n;

    do

    {

        printf("Enter number as your operation:\n1 -----> Enqueue\n2 -----> Dequeue\n3 -----> Display\n0 -----> Exit\n");

        scanf("%d",&o);

        switch(o)

        {

        case 1:

            enqueue(&q);

            break;

        case 2:

            n=dequeue(&q);

            if(n!=0)

                printf("The Dequeued number: %d\n",n);

            break;

        case 3:

            display(q);

            break;

        default:

            if(o!=0)

                printf("Wrong input\n");

            break;

        }

    }

    while (o!=0);

    return 0;

}

void enqueue(struct queue \*q)

{

    int n;

    if((((q->rear)+1)%MaxSize)==q->front)

        printf("Queue is full\n");

    else

    {

        printf("Enter the element to enqueue: ");

        scanf("%d",&n);

        if((q->front==-1)&&(q->rear==-1))

        {

            q->front=0;

            q->rear=0;

            q->arr[q->rear]=n;

        }

        else

        {

            q->rear=((q->rear)+1)%MaxSize;

            q->arr[q->rear]=n;

        }

    }

}

int dequeue(struct queue \*q)

{

    int item;

    if((q->front==-1)&&(q->rear==-1))

    {

        printf("Queue is empty\n");

        return 0;

    }

    else if((q->front)==(q->rear))

    {

        item=q->arr[q->front];

        q->front=-1;

        q->rear=-1;

        return item;

    }

    else

    {

        item=q->arr[q->front];

        q->front=((q->front)+1)%MaxSize;

        return item;

    }

}

void display(struct queue q)

{

    int i=q.front;

    if((q.front==-1)&&(q.rear==-1))

        printf("Queue is empty\n");

    else

    {

        printf("Queue elements: ");

        while(i!=q.rear)

        {

            printf("%d ",q.arr[i]);

            i=(i+1)%MaxSize;

        }

        printf("%d\n",q.arr[q.rear]);

    }

}

**Question 2**

A Barua number is a number which consists of only zeroes and ones and has only one 1. Barua number will start with 1. Given numbers, find out the multiplication of the numbers. Note: The input may contain one decimal number and all other Barua numbers. (Assume that each number is very large and total number of values give is also very large)

Input 1: 100 10 12 1000

Output 1: 12000000

Input 2: 100 121 1000000000000000

Output 2: 12100000000000000000

Input 3: 10 100 1000

Output 3: 1000000

Function code:

void barua (long a[], int n)

{

int count=0,i;

long x,p=1;

for(i=0;i<n;i++)

{

a[i]=x;

while(x%10==0)

{

count++;

x=x/10;

}

p=p\*x;

}

printf(“Output: %d“,p);

for(i=1;i<=count;i++)

{

printf(“0”);

}

}

**Question 3**

Implement push, pop and find the minimum element in a stack in O(1) time complexity.

Program code:

#include<stdio.h>

#define MaxSize 5

struct stack

{

    int arr[MaxSize];

    int top;

};

void push(struct stack \*s, struct stack \*minstack);

int pop(struct stack \*s, struct stack \*minstack);

void display(struct stack s);

int min(struct stack minstack);

int main()

{

    struct stack s,minstack;

    s.top=-1;

    int num,o,min;

    do

    {

        printf("Enter number as your operation:\n1 -----> Push\n2 -----> Pop\n3 -----> Display\n4 -----> Find Minimum\n0 -----> Exit\n");

        scanf("%d",&o);

        switch(o)

        {

        case 1:

            push(&s,&minstack);

            break;

        case 2:

            num=pop(&s,&minstack);

            if(num!=0)

                printf("The Popped number: %d\n",num);

            break;

        case 3:

            display(s);

            break;

case 4:

min=min(minstack);

printf(“Minimun element: %d\n”,min);

break;

        default:

            if(o!=0)

                printf("Wrong input\n");

            break;

        }

    }

    while(o!=0);

    return 0;

}

void push(struct stack \*s)

{

    int n1;

    if(s->top==MaxSize-1)

        printf("Stack is full\n");

    else

    {

        printf("Enter element to push: ");

        scanf("%d",&n1);

        s->top=s->top+1;

minstack->top=minstack->top+1;

        s->arr[s->top]=n1;

if(n1 < minstack->arr[minstack->(top-1)])

minstack->arr[minstack->top]=n1;

else

minstack->arr[minstack->top]= minstack->arr[minstack->(top-1)];

    }

}

int pop(struct stack \*s)

{

    int item;

    if(s->top==-1)

    {

        printf("Stack is Empty\n");

        return 0;

    }

    else

    {

        item=s->arr[s->top];

        s->top=s->top-1;

minstack->top=minstack->top-1;

        return item;

    }

}

void display(struct stack s)

{

    int i;

    if(s.top==-1)

        printf("Stack is empty\n");

    else

    {

        printf("Stack elements: ");

        for(i=s.top;i>=0;i--)

        {

            printf("%d ",s.arr[i]);

        }

        printf("\n");

    }

}

int min(struct stack minstack)

{

int x= minstack->arr[minstack->top];

return x;

}