QUESTION 1 ANSWER :

// Circular Queue implementation

#include <stdio.h>

#define SIZE 5

int items[SIZE];

int front = -1, rear = -1;

// Check if the queue is full

int isFull() {

if ((front == rear + 1) || (front == 0 && rear == SIZE - 1)) return 1;

return 0;

}

// Check if the queue is empty

int isEmpty() {

if (front == -1) return 1;

return 0;

}

// Adding an element

void enQueue(int element) {

if (isFull())

printf("\n Queue is full!! \n");

else {

if (front == -1) front = 0;

rear = (rear + 1) % SIZE;

items[rear] = element;

printf("\n Inserted -> %d", element);

}

}

// Removing an element

int deQueue() {

int element;

if (isEmpty()) {

printf("\n Queue is empty !! \n");

return (-1);

} else {

element = items[front];

if (front == rear) {

front = -1;

rear = -1;

}

// Q has only one element, so we reset the

// queue after dequeing it. ?

else {

front = (front + 1) % SIZE;

}

printf("\n Deleted element -> %d \n", element);

return (element);

}

}

// Display the queue

void display() {

int i;

if (isEmpty())

printf(" \n Empty Queue\n");

else {

printf("\n Front -> %d ", front);

printf("\n Items -> ");

for (i = front; i != rear; i = (i + 1) % SIZE) {

printf("%d ", items[i]);

}

printf("%d ", items[i]);

printf("\n Rear -> %d \n", rear);

}

}

int main() {

// Fails because front = -1

deQueue();

enQueue(1);

enQueue(2);

enQueue(3);

enQueue(4);

enQueue(5);

// Fails to enqueue because front == 0 && rear == SIZE - 1

enQueue(6);

display();

deQueue();

display();

enQueue(7);

display();

// Fails to enqueue because front == rear + 1

enQueue(8);

return 0;

}

QUESTION 2 ANSWER :

//BARUA NUMBER

#include <stdio.h>

int main()

{ int a1[4]={100, 10 ,12 ,1000 };

int count=0, p;

p=1;

for(int i =0;i<4;i++)

{ if (a1[i]%10!=0)

{

p=p\*a1[i];}

while(a1[i]%10==0)

{

count++;

a1[i]=a1[i]/10;

}

}

printf("%d",p);

for (int j=1;j<=count;j++)

printf ("0");

return 0;

}

//output 1: 12000000

#include <stdio.h>

int main()

{ int a1[3]={100,121,1000000000000000};

int count=0, p;

p=1;

for(int i =0;i<3;i++)

{ if (a1[i]%10!=0)

{

p=p\*a1[i];}

while(a1[i]%10==0)

{

count++;

a1[i]=a1[i]/10;

}

}

printf("%d",p);

for (int j=1;j<=count;j++)

printf ("0");

return 0;

}

//output 2: 12100000000000000000

#include <stdio.h>

int main()

{ int a1[3]={10,100,1000};

int count=0, p;

p=1;

for(int i =0;i<3;i++)

{ if (a1[i]%10!=0)

{

p=p\*a1[i];}

while(a1[i]%10==0)

{

count++;

a1[i]=a1[i]/10;

}

}

printf("%d",p);

for (int j=1;j<=count;j++)

printf ("0");

return 0;

}

//output 3:1000000

QUESTION 3 ANSWER :

#include <stdio.h>

#include <stdlib.h>

int top=-1 ,n ,stk[10],sstk[10];

int min;

void push();

void pop();

void display();

void min\_ele();

void main(){

int ch;

printf("enter size of stack \n");

scanf("%d",&n );

printf("\n MENU \n");

printf("1.PUSH\n2.POP\n3.DISPLAY\n4.minimum\_element\n5.EXIT");

do

{

printf("enter your choice\n");

scanf("%d",&ch);

switch (ch)

{

case 1:push();

break;

case 2:pop();

break;

case 3:display();

break;

case 4: min\_ele();

break;

case 5:exit(1);

break;

default:printf("Enter correct Charecter\n");

break;

}

}

while ( ch!=5 );

}

void push(){

int x,temp;

if(top == n-1){

printf("Stack Overflow\n");

}

else

{

top++;

printf("\nenter the value\n");

scanf("%d",&x);

stk[top]=x;

sstk[top]=x;

if (top>0&&sstk[top-1]<sstk[top])

{

temp=sstk[top];

sstk[top]=sstk[top-1];

sstk[top-1]=temp;

}

}

}

void pop(){

if(top == -1){

printf(";nStack Underflow\n");

}

else

{

printf("\n%d is deletedd \n",stk[top]);

top--;

}

}

void min\_ele(){

if (top <0)

{

printf ("\n stack is empty \n");

}

else

printf("\nThe minimum element is %d\n",sstk[top]);}

void display()

{

if(top == -1){

printf("Stack is empty\n");

}

else

{

for(int i = top;i>=0;i--){

printf("%3d\t",stk[i]);

}

}

}