Day 7

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

stack circularqueue{

public:

int capacity;

int \*arr;

int f;

int r;

};

/////// create queue

Stack circularqueue\* createqueue(int cap){

Stack circularqueue \*newqueue = (stack circularqueue \*) malloc(sizeof(stack circularqueue ));

newqueue->f = 0;

newqueue->r = 0;

newqueue->capacity = cap;

newqueue->arr = new int(newqueue->capacity);

return newqueue;

}

///// check queue is empty or not

int isempty(circularqueue \*q){

if(q->f == q->r){

return 1;

}

return 0;

}

///// check queue is full or not

int isfull(circularqueue \*q){

if((q->r+1) % q->capacity == q->f){

return 1;

}

return 0;

}

///// insert in queue

void enqueue(circularqueue \*q,int val){

if(isfull(q)){

printf("queue is full \n");

}

else{

q->r = (q->r+1)%q->capacity;

q->arr[q->r] = val;

}

}

///// delete in queue

void dqueue(circularqueue \*q){

if(isempty(q)){

printf("queue is empty \n");

return;

}else{

q->f = (q->f+1) % q->capacity;

int pop = q->arr[q->f];

printf("Pop element is: \n “);

}

}

///// display queue

void display(circularqueue \*q) {

int i;

if (isempty(q))

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

else {

for (i = q->f; i != q->r; i = (i + 1) % q->capacity) {

printf("%d ", q->arr[i]);

}

printf("%d ", q->arr[i]);

}

}

main(){

circularqueue \*q = createqueue(3);

enqueue(q,10);

enqueue(q,20);

dqueue(q);

enqueue(q,30);

}