DSA Experiment 5

Harshit Saraswat | 500097101 | B42

**QUESTION 1:** Using array and functions implement Circular Queue data structure and its operations like insert, delete, and display.

**CODE:**

#include <stdio.h>

#include <stdlib.h>

int count = 0;

int \* queue;

int front = -1;

int rear = -1;

int size = 0;

int main()

{

printf("Enter Size of Queue : ");

scanf("%d",&size);

queue = (int\*)malloc(size\*sizeof(int));

int choice;

while (1 == 1){

printf("1 - Enqueue\n2 - Dequeue\n3 - Print Queue\n4 - Print Info\n5 - Quit\n\nChoice : ");

scanf("%d",&choice);

printf("\n");

switch(choice){

case 1 : enqueue();break;

case 2 : dequeue();break;

case 3 : print\_queue();break;

case 4 : info();break;

case 5 : exit(0);

}

printf("\n");

}

return 0;

}

void enqueue(){

if (front == -1){

front++;

rear++;

count++;

printf("Enter Element to Enqueue : ");

scanf("%d",&queue[rear]);

return;

}

if (count == size){

printf("Queue is FULL... Pls Dequeue to Continue\n");

return;

}

count++;

rear = (rear+1) % size;

printf("Enter Element to Enqueue : ");

scanf("%d",&queue[rear]);

}

void dequeue(){

if (count == 0){

printf("Queue is EMPTY... Pls Enqueue something first\n");

}

count--;

printf("Dequeued Element = %d\n",queue[front]);

front = (front+1)%size;

}

void print\_queue(){

int temp\_count = 0;

int temp\_front = front;

printf("\n");

for (temp\_count; temp\_count!=count ; temp\_count++,temp\_front++){

printf("| %d |", queue[temp\_front]);

}

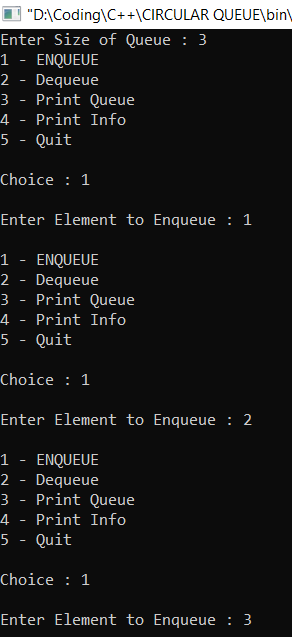
}

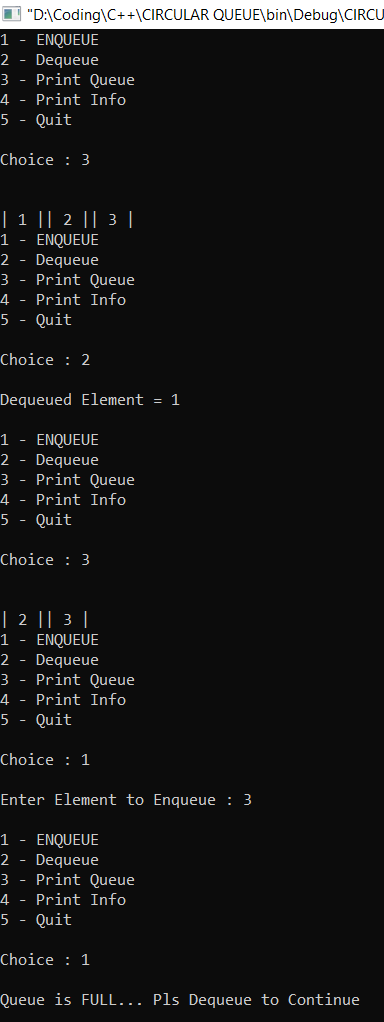
void info(){

printf("\nCount is %d Front is %d Rear is %d\n",count,front,rear);

}

**OUTPUT:**





**QUESTION 2 :** Using array and functions implement Priority Queue and its operations (insert, delete, display)

**CODE :**

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

int priority;

};

struct node \* queue;

int size = 0;

int count = 0;

int main()

{

printf("Enter Size of Queue : ");

scanf("%d",&size);

queue = (struct node \*)malloc(size\*sizeof(struct node));

int choice = 1;

while (choice!=4){

printf("\nTo Insert - 1\nTo Delete - 2\nTo Display - 3\nExit - 4\n\nCHOICE : ");

scanf("%d",&choice);

switch(choice){

case 1 : insert(); break;

case 2 : deleter(); break;

case 3 : display(); break;

case 4 : exit(0);

}

}

return 0;

}

void insert(){

if (size == count){ //QUEUE IS FULL

printf("\nQueue FULL... Exiting");

return;

}

count++;

printf("Enter Data to Insert : ");

scanf("%d",&queue[count - 1].data);

printf("Enter Priority of the Data : ");

scanf("%d",&queue[count - 1].priority);

if (count == 1){ //NO NEED TO SORT THROUGH WHEN NO OTHER ELEMENTS TO COMPARE WITH

return;

}

int position\_of\_node = count - 1; //

while(queue[position\_of\_node].priority < queue[position\_of\_node - 1].priority && position\_of\_node != 0){ //TILL THE NODE IS NOT AT THE PERFECT PRIORITY KEEP GOING OR TILL AT INDEX 0

swap\_in\_queue(position\_of\_node,position\_of\_node - 1);

position\_of\_node--;

}

}

void deleter(){

printf("Dequeued Element : %d",queue[0]);

count--;

for (int i = 0; i!=count;i++){

swap\_in\_queue(i,i+1);

}

}

void swap\_in\_queue(int a, int b){

struct node temp = queue[a];

queue[a] = queue[b];

queue[b] = temp;

}

void display(){

printf("\nThe Queue is : \n");

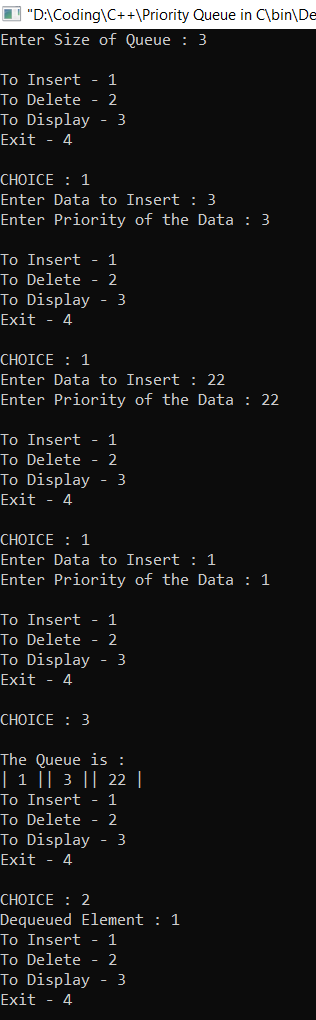
for (int i = 0; i != count; i++){

printf("| %d |",queue[i]);

}

}

**OUTPUT:**

****

**QUESTION 3:** Using linked list implement Double Ended Queue (Input Restricted Deque and Output Restricted Deque) and its operations (insert, delete, display).

**CODE:**

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \* next;

};

struct node \* front;

struct node \* rear;

struct node \* new\_node;

struct node \* ptr;

void insert\_at\_front();

void insert\_at\_rear();

void delete\_at\_front();

void delte\_at\_rear();

int main()

{

printf("For Input Restricted Deque - 1\nFor Output Restricted Deque - 2\n");

int type = 0;

scanf("%d",&type);

switch (type){

case 1 : printf("\t\t\t\tINPUT RESTRICTED QUEUE\n\n");InpR\_Deque();break;

case 2 : printf("\t\t\t\tOUTPUT RESTRICTED QUEUE\n\n");OutR\_Deque();break;

}

return 0;

}

void InpR\_Deque(){

while(1==1){

int choice = 0;

printf("\n1 - Input At Front\n2 - Delete at Front\n3 - Delete at Rear\n4 - Display\n5 - Quit\n\nChoice : ");

scanf("%d",&choice);

printf("\n");

switch (choice){

case 1 : insert\_at\_front();break;

case 2 : delete\_at\_front();break;

case 3 : delete\_at\_rear();break;

case 4 : display\_queue();break;

case 5 : exit(0);

}

}

}

void OutR\_Deque(){

while(1==1){

int choice = 0;

printf("1 - Input At Front\n2 - Input at rear\n3 - Delete at Rear\n\n4 - Display\n5 - Quit\nChoice : ");

scanf("%d",&choice);

switch (choice){

case 1 : insert\_at\_front();break;

case 2 : insert\_at\_rear();break;

case 3 : delete\_at\_rear();break;

case 4 : display\_queue();break;

case 5 : exit(0);

}

}

}

void info(){

printf("FRONT = %d REAR = %d FRONT->NEXT = %d REAR->NEXT = %d",front,rear,front->next,rear->next);

}

void insert\_at\_front(){

new\_node = (struct node \*)malloc(sizeof(struct node));

printf("Enter Data To Insert at Front : ");

scanf("%d",&new\_node->data);

if (front == NULL){

new\_node->next = NULL;

front = new\_node;

rear = new\_node;

display\_queue();

return;

}

new\_node->next = front;

front = new\_node;

display\_queue();

}

void insert\_at\_rear(){

new\_node = (struct node \*)malloc(sizeof(struct node));

printf("Enter Data To Insert at Rear : ");

scanf("%d",&new\_node->data);

new\_node->next = NULL;

if (front == NULL){

front = rear = new\_node;

display\_queue();

return;

}

rear->next = new\_node;

rear = new\_node;

display\_queue();

}

void delete\_at\_front(){

if (front == NULL){

printf("\nQUEUE IS EMPTY...");

return;

}

if (front == rear){

printf("Deleted Element at front : %d\n",front->data);

free(front);

front = rear = NULL;

display\_queue();

return;

}

printf("Deleted Element at front : %d\n",front->data);

struct node \* temp = front;

front = front->next;

free(temp);

display\_queue();

}

void delete\_at\_rear(){

if (front == NULL){

printf("\nQUEUE IS EMPTY...");

return;

}

if (front == rear){

printf("Deleted Element at rear : %d\n",front->data);

free(front);

front = rear = NULL;

display\_queue();

return;

}

printf("Deleted Element at rear : %d\n",front->data);

for (ptr = front; ptr->next!=rear; ptr = ptr->next);

struct node \* temp = rear;

rear = ptr;

rear->next = NULL;

free(temp);

display\_queue();

}

void display\_queue(){

printf("\n");

if (front == NULL){

printf("QUEUE IS EMPTY...\n");

return;

}

for(ptr = front; ptr!=rear->next;ptr=ptr->next){

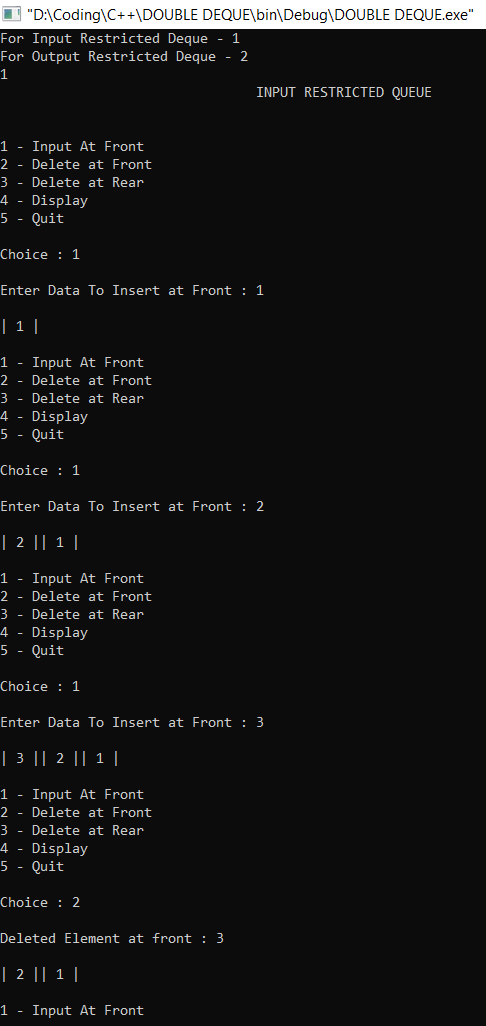
printf("| %d |",ptr->data);

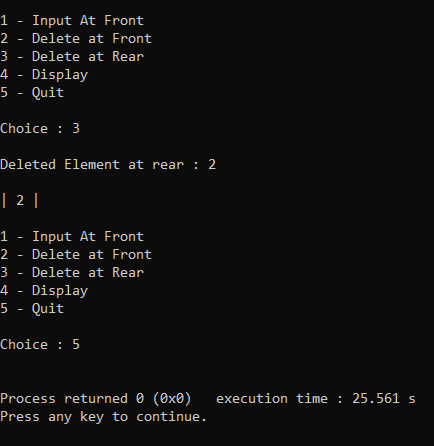
}

printf("\n");

}

**OUTPUT :**





THANK YOU!