**VARUN KUMAR**

**2K19-IT-140**

**DS LAB – 5**

9 .Write a program to implement Queue Data structure

// C program for array implementation of queue

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

// A structure to represent a queue

struct Queue {

int front, rear, size;

unsigned capacity;

int\* array;

};

// function to create a queue

// of given capacity.

// It initializes size of queue as 0

struct Queue\* createQueue(unsigned capacity)

{

struct Queue\* queue = (struct Queue\*)malloc(

sizeof(struct Queue));

queue->capacity = capacity;

queue->front = queue->size = 0;

// This is important, see the enqueue

queue->rear = capacity - 1;

queue->array = (int\*)malloc(

queue->capacity \* sizeof(int));

return queue;

}

// Queue is full when size becomes

// equal to the capacity

int isFull(struct Queue\* queue)

{

return (queue->size == queue->capacity);

}

// Queue is empty when size is 0

int isEmpty(struct Queue\* queue)

{

return (queue->size == 0);

}

// Function to add an item to the queue.

// It changes rear and size

void enqueue(struct Queue\* queue, int item)

{

if (isFull(queue))

return;

queue->rear = (queue->rear + 1)

% queue->capacity;

queue->array[queue->rear] = item;

queue->size = queue->size + 1;

printf("%d enqueued to queue\n", item);

}

// Function to remove an item from queue.

// It changes front and size

int dequeue(struct Queue\* queue)

{

if (isEmpty(queue))

return INT\_MIN;

int item = queue->array[queue->front];

queue->front = (queue->front + 1)

% queue->capacity;

queue->size = queue->size - 1;

return item;

}

// Function to get front of queue

int front(struct Queue\* queue)

{

if (isEmpty(queue))

return INT\_MIN;

return queue->array[queue->front];

}

// Function to get rear of queue

int rear(struct Queue\* queue)

{

if (isEmpty(queue))

return INT\_MIN;

return queue->array[queue->rear];

}

// Driver program to test above functions./

int main()

{

struct Queue\* queue = createQueue(100);

enqueue(queue, 50);

enqueue(queue, 14);

enqueue(queue, 26);

enqueue(queue, 30);

enqueue(queue, 45);

enqueue(queue, 100);

printf("\n%d dequeued from queue\n\n",

dequeue(queue));

printf("%d dequeued from queue\n\n",

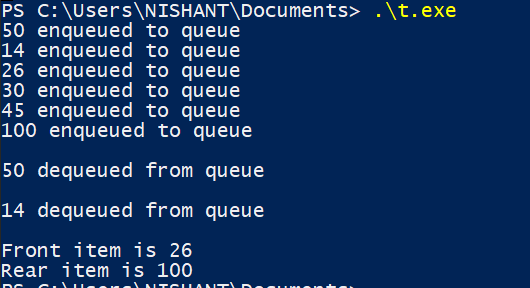
dequeue(queue));

printf("Front item is %d\n", front(queue));

printf("Rear item is %d\n", rear(queue));

return 0;

}



# 10: Write a program to reverse first k elements of a given Queue.

**#include <stdio.h>**

**int rear=-1,front=-1,a[20];**

**int x;**

**void enqueue()**

**{**

**if(rear==x-1)**

**printf("NOTE: STACK IS FULL\n");**

**else if(rear==-1&&front==-1)**

**{**

**rear=front=0;**

**printf("ENTER THE NO. TO BE ADDED: ");**

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

**}**

**else**

**{**

**rear++;**

**printf("ENTER THE NO. TO BE ADDED: ");**

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

**}**

**}**

**void dequeue()**

**{**

**if(rear==-1&&front==-1)**

**printf("NOTE: STACK IS EMPTY\n");**

**else if(rear==front)**

**{**

**printf("NO. TO BE REMOVED IS: %d\n",a[front]);**

**rear=front=-1;**

**}**

**else**

**{**

**printf("NO. TO BE REMOVED IS: %d\n",a[front]);**

**front++;**

**}**

**}**

**void peek()**

**{**

**if(front==-1)**

**printf("NOTE: STACK IS EMPTY\n");**

**else**

**{**

**printf("NO. AT FRONT IS: %d\n",a[front]);**

**}**

**}**

**void display()**

**{**

**if(front==-1)**

**printf("NOTE: STACK IS EMPTY\n");**

**else**

**{**

**int i;**

**printf("ELEMENTS IN THE QUEUE ARE\n");**

**for(i=front;i<=rear;i++)**

**printf("%d ",a[i]);**

**printf("\n");**

**}**

**}**

**void reverse()**

**{**

**if(front==-1)**

**printf("NOTE: STACK IS EMPTY\n");**

**else**

**{**

**int t;**

**printf("Enter the no. of elements to be reversed: ");**

**scanf("%d",&t);**

**if(t>rear-front+1)**

**printf("ERROR\n");**

**else**

**{**

**int b[t],i,j;**

**for(i=front,j=0;j<t;j++)**

**{**

**b[j]=a[i];**

**i++;**

**}**

**for(i=front,j=t-1;j>=0;j--)**

**{**

**a[i]=b[j];**

**i++;**

**}**

**printf("Queue after reversing %d elements\n",x);**

**display();**

**}**

**}**

**}**

**int main() {**

**int n;**

**printf("Enter the size of QUEUE: ");**

**scanf("%d",&x);**

**printf("PRESS\n");**

**do**

**{**

**printf("1)ENQUEUE 2)DEQUEUE 3)PEEK 4)DISPLAY 5)REVERSE 6)EXIT\n");**

**printf("Enter your CHOICE: ");**

**scanf("%d",&n);**

**switch(n)**

**{**

**case 1:**

**{**

**enqueue();**

**break;**

**}**

**case 2:**

**{**

**dequeue();**

**break;**

**}**

**case 3:**

**{**

**peek();**

**break;**

**}**

**case 4:**

**{**

**display();**

**break;**

**}**

**case 5:**

**{**

**reverse();**

**break;**

**}**

**case 6:**

**{**

**printf("\*!Wrong!\*\n");**

**break;**

**}**

**default:**

**{**

**printf("NOTE: ENTER A VALID CHOICE\n");**

**break;**

**}**

**}**

**}**

**while(n!=5);**

**return 0;**

**}**

