DATA STRUCTURES AND ALOGRITHM

Saravanakaruppu. K

BATCH ONE

QUESTION : write a program for implementing the insert, deleting and display operation in circular queue

/ C or C++ program for insertion and

// deletion in Circular Queue

#include<bits/stdc++.h>

Using namespace std;

Struct Queue

{

// Initialize front and rear

Int rear, front;

// Circular Queue

Int size;

Int \*arr;

Queue(int s)

{

Front = rear = -1;

Size = s;

Arr = new int[s];

}

Void enQueue(int value);

Int deQueue();

Void displayQueue();

};

/\* Function to create Circular queue \*/

Void Queue::enQueue(int value)

{

If ((front == 0 && rear == size-1) ||

(rear == (front-1)%(size-1)))

{

Printf(“\nQueue is Full”);

Return;

}

Else if (front == -1) /\* Insert First Element \*/

{

Front = rear = 0;

Arr[rear] = value;

}

Else if (rear == size-1 && front != 0)

{

Rear = 0;

Arr[rear] = value;

}

Else

{

Rear++;

Arr[rear] = value;

}

}

// Function to delete element from Circular Queue

Int Queue::deQueue()

{

If (front == -1)

{

Printf(“\nQueue is Empty”);

Return INT\_MIN;

}

Int data = arr[front];

Arr[front] = -1;

If (front == rear)

{

Front = -1;

Rear = -1;

}

Else if (front == size-1)

Front = 0;

Else

Front++;

Return data;

}

// Function displaying the elements

// of Circular Queue

Void Queue::displayQueue()

{

If (front == -1)

{

Printf(“\nQueue is Empty”);

Return;

}

Printf(“\nElements in Circular Queue are: “);

If (rear >= front)

{

For (int I = front; I <= rear; i++)

Printf(“%d “,arr[i]);

}

Else

{

For (int I = front; I < size; i++)

Printf(“%d “, arr[i]);

For (int I = 0; I <= rear; i++)

Printf(“%d “, arr[i]);

}

}

/\* Driver of the program \*/

Int main()

{

Queue q(5);

// Inserting elements in Circular Queue

q.enQueue(14);

q.enQueue(22);

q.enQueue(13);

q.enQueue(-6);

// Display elements present in Circular Queue

q.displayQueue();

// Deleting elements from Circular Queue

Printf(“\nDeleted value = %d”, q.deQueue());

Printf(“\nDeleted value = %d”, q.deQueue());

q.displayQueue();

q.enQueue(9);

q.enQueue(20);

q.enQueue(5);

q.displayQueue();

q.enQueue(20);

return 0;

}