DAY5 LAB

1. Write a program that implement Queue (its operations) using Arrays.

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int queue[MAX\_SIZE];

int front = -1, rear = -1;

void enqueue(int value) {

if (rear == MAX\_SIZE - 1) {

printf("Queue is full. Cannot enqueue element.\n");

} else {

if (front == -1) {

front = 0;

}

rear++;

queue[rear] = value;

printf("%d enqueued to the queue.\n", value);

}

}

void dequeue() {

if (front == -1) {

printf("Queue is empty. Cannot dequeue element.\n");

} else {

printf("%d dequeued from the queue.\n", queue[front]);

if (front == rear) {

front = rear = -1;

} else {

front++;

}

}

}

void display() {

if (front == -1) {

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

} else {

printf("Queue elements are: ");

for (int i = front; i <= rear; i++) {

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

}

printf("\n");

}

}

int main() {

enqueue(10);

enqueue(20);

enqueue(30);

display();

dequeue();

dequeue();

display();

return 0;

}

OUTPUT:

10 enqueued to the queue.

20 enqueued to the queue.

30 enqueued to the queue.

Queue elements are: 10 20 30

10 dequeued from the queue.

20 dequeued from the queue.

Queue elements are: 30

1. Write a program that implement Queue (its operations) using Linked list(Pointers).

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Queue {

struct Node \*front, \*rear;

};

struct Node\* newNode(int data) {

struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));

temp->data = data;

temp->next = NULL;

return temp;

}

struct Queue\* createQueue() {

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

queue->front = queue->rear = NULL;

return queue;

}

void enqueue(struct Queue\* queue, int data) {

struct Node\* temp = newNode(data);

if (queue->rear == NULL) {

queue->front = queue->rear = temp;

return;

}

queue->rear->next = temp;

queue->rear = temp;

}

void dequeue(struct Queue\* queue) {

if (queue->front == NULL)

return;

struct Node\* temp = queue->front;

queue->front = queue->front->next;

if (queue->front == NULL)

queue->rear = NULL;

free(temp);

}

int main() {

struct Queue\* queue = createQueue();

enqueue(queue, 10);

enqueue(queue, 20);

enqueue(queue, 30);

dequeue(queue);

printf("Queue Front: %d\n", queue->front->data);

printf("Queue Rear: %d\n", queue->rear->data);

return 0;

}

OUTPUT:

Queue Front: 20

Queue Rear: 30