DS day5

1.Write a program that implement Queue (its operations) using Arrays. #include <stdio.h>

#include <stdlib.h>

#define MAX 5

struct Queue {

int items[MAX];

int front;

int rear;

};

struct Queue\* createQueue() {

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

q->front = -1;

q->rear = -1;

return q;

}

int isFull(struct Queue\* q) {

return q->rear == MAX - 1;

}

int isEmpty(struct Queue\* q) {

return q->front == -1 || q->front > q->rear;

}

void enqueue(struct Queue\* q, int value) {

if (isFull(q)) {

printf("Queue is full!\n");

return;

}

if (q->front == -1) {

q->front = 0;

}

q->rear++;

q->items[q->rear] = value;

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

}

int dequeue(struct Queue\* q) {

if (isEmpty(q)) {

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

return -1;

}

int item = q->items[q->front];

q->front++;

if (q->front > q->rear) {

q->front = q->rear = -1;

}

return item;

}

void display(struct Queue\* q) {

if (isEmpty(q)) {

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

return;

}

printf("Queue elements: ");

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

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

}

printf("\n");

}

int main() {

struct Queue\* q = createQueue();

enqueue(q, 10);

enqueue(q, 20);

enqueue(q, 30);

display(q);

printf("%d dequeued from queue\n", dequeue(q));

display(q);

enqueue(q, 40);

enqueue(q, 50);

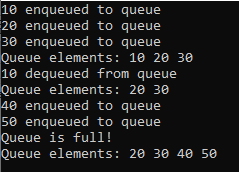
enqueue(q, 60);

display(q);

return 0;

}

Output:



2.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("Front element: %d\n", queue->front->data);

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

return 0;

}

Output:

