Day 5

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) {
```

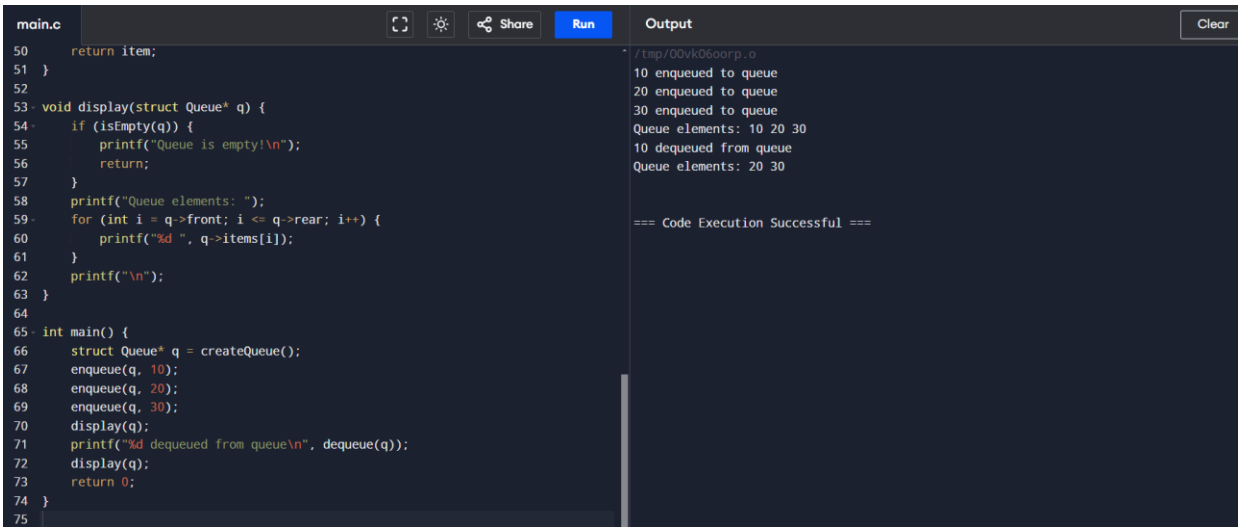
```
        q->front = q->rear = -1;
    }
    return item;
}

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);
```

```
    return 0;
}
```

Output:



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

```
#include <stdio.h>
#include <stdlib.h>

// Node structure
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Queue structure
```

```
typedef struct Queue {
    Node* front;
    Node* rear;
} Queue;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to create a new queue
Queue* createQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->front = NULL;
    q->rear = NULL;
    return q;
}

// Function to enqueue an element
void enqueue(Queue* q, int data) {
    Node* newNode = createNode(data);
    if (q->rear == NULL) {
```

```
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
}

// Function to dequeue an element
int dequeue(Queue* q) {
    if (q->front == NULL) {
        printf("Queue is empty\n");
        return -1;
    } else {
        Node* temp = q->front;
        int data = temp->data;
        q->front = q->front->next;
        if (q->front == NULL) {
            q->rear = NULL;
        }
        free(temp);
        return data;
    }
}

// Function to print the queue
```

```
void printQueue(Queue* q) {
	Node* temp = q->front;
	while (temp != NULL) {
		printf("%d ", temp->data);
		temp = temp->next;
	}
	printf("\n");
}

int main() {
	Queue* q = createQueue();

	enqueue(q, 1);
	enqueue(q, 2);
	enqueue(q, 3);
	enqueue(q, 4);
	enqueue(q, 5);

	printQueue(q); // Output: 1 2 3 4 5

	int dequeued = dequeue(q);
	printf("Dequeued: %d\n", dequeued); // Output: 1

	printQueue(q); // Output: 2 3 4 5
```

```
    return 0;
}
```

output: