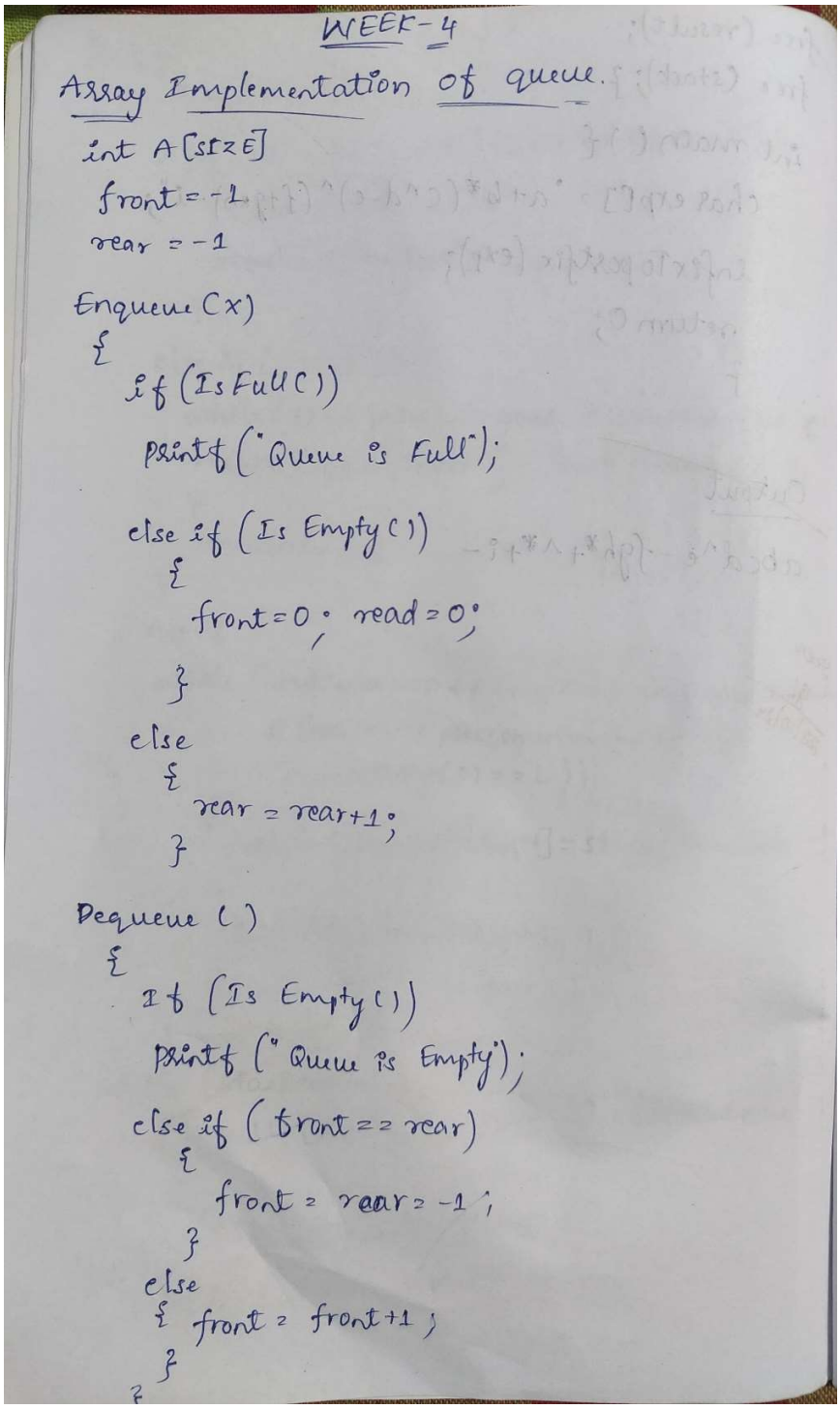
Program 3

WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display
The program should print appropriate messages for queue empty and queue overflow conditions.

Observation:



WEEK-4

Array Implementation of queue.

```
int A[SIZE]
front = -1
rear = -1

Enqueue(x)
{
  if (IsFull())
   printf("Queue is Full");

  else if (Is Empty())
    {
      front=0; read=0;
    }
  else
    {
      rear = rear+1;
    }

Dequeue()
{
  If (Is Empty())
   printf("Queue is Empty");
  else if (front==rear)
    {
      front = rear = -1;
    }
  else
   { front = front+1;
   }
}
```

```
IsFull()
{
  if (rear == SIZE-1)
    return true;
  else
    return false;
}
IsEmpty()
{
  if (front == -1 && rear == -1)
    return True;
  else
    return false;
}
```

Seen

```
void display(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")
}
```

Output

```
Enqueued 10
Enqueued 20
Enqueued 30
Queue elements : 10 20 30
Dequeued 10
Queue elements : 20 30
Enqueued 40
Enqueued 50
Queue Elements : 20 30 40 50
Dequeued 20
Dequeued 30
Dequeued 40
Dequeued 50
Queue is empty
```

Seen

Seen

14/10/24

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int isFull(int rear) {
    if (rear == MAX - 1) {
        return 1;
    }
    return 0;
}

int isEmpty(int front, int rear) {
    if (front == -1 || front > rear) {
        return 1;
    }
    return 0;
}

void insert(int queue[], int *front, int *rear, int value) {
    if (isFull(*rear)) {
        printf("Queue Overflow! Cannot insert %d\n", value);
        return;
    }

    if (*front == -1) {
        *front = 0;
    }

    (*rear)++;
    queue[*rear] = value;
    printf("%d inserted into the queue\n", value);
}

void delete(int queue[], int *front, int *rear) {
    if (isEmpty(*front, *rear)) {
        printf("Queue Underflow! No element to delete\n");
        return;
    }

    int deletedValue = queue[*front];
    printf("%d deleted from the queue\n", deletedValue);
    (*front)++;

    // Reset the queue if it becomes empty
    if (*front > *rear) {
        *front = *rear = -1;
    }
```

```c
}

void display(int queue[], int front, int rear) {
    if (isEmpty(front, rear)) {
        printf("Queue is empty!\n");
        return;
    }

    printf("Queue elements: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int queue[MAX];
    int front = -1, rear = -1;

    int choice, value;

    while (1) {
        printf("\nQueue Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to insert: ");
                scanf("%d", &value);
                insert(queue, &front, &rear, value);
                break;

            case 2:
                delete(queue, &front, &rear);
                break;

            case 3:
                display(queue, front, rear);
                break;

            case 4:
                exit(0);

            default:
                printf("Invalid choice! Please try again.\n");
```

```
        }
    }

    return 0;
}
```

Output:

```
C:\Users\satis\practice>gcc Program3.c

C:\Users\satis\practice>a.exe

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 5
5 inserted into the queue

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 7
7 inserted into the queue

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 9
9 inserted into the queue

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to insert: 10
10 inserted into the queue
```

```
Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 5 7 9 10

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
5 deleted from the queue

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
7 deleted from the queue

Queue Operations:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 9 10
```