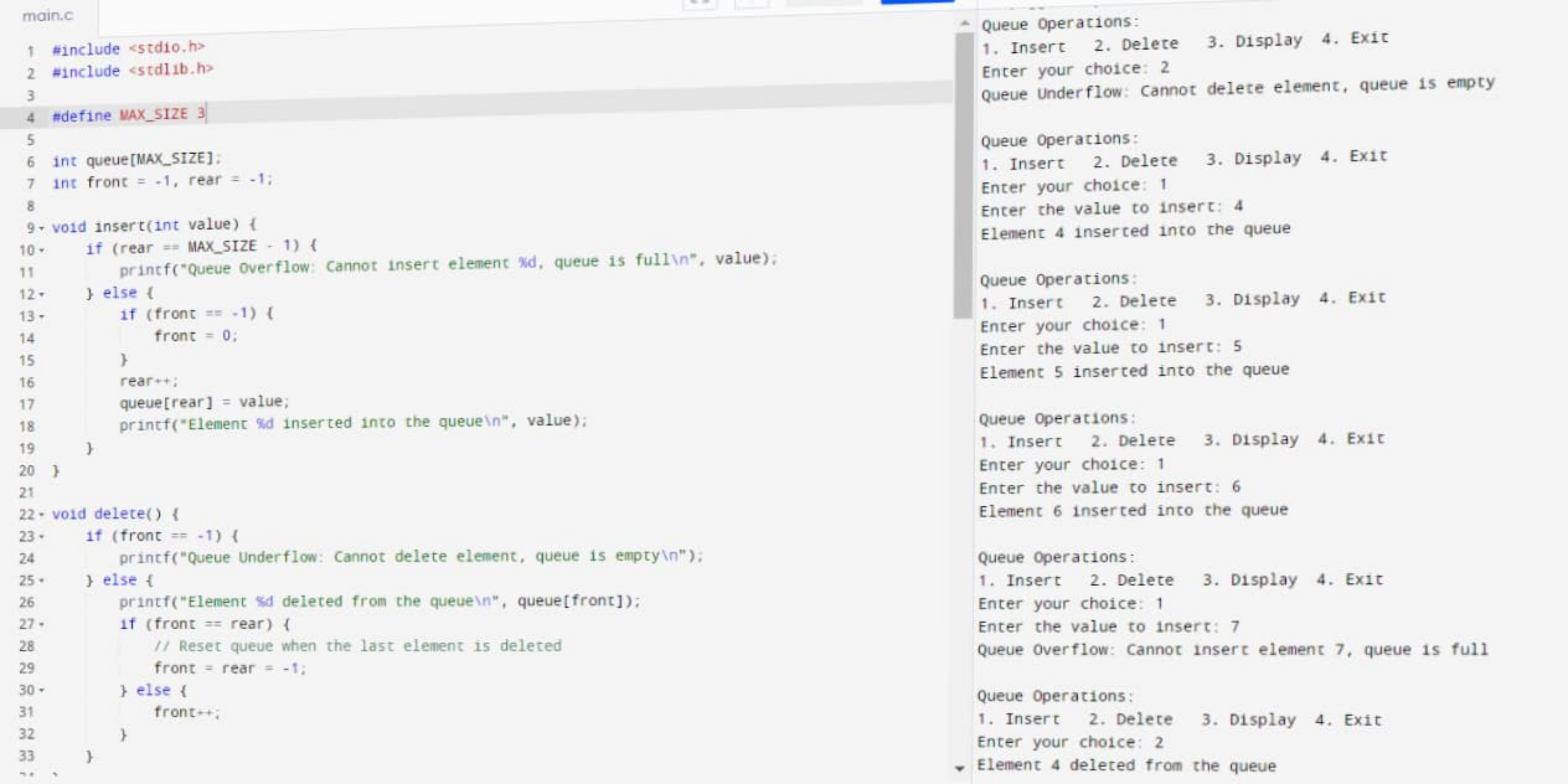
LAB - 3

1. LINEAR QUEUE USING ARRAYS.

```
#include<stdio.h>
~~#include<c~~
#define MAX 10
int queue[MAX], front=-1, rear=-1;
void insert(){
    int num;
    printf("Enter number to be inserted: ");
    scanf("%d", &num);
    if (rear==MAX-1){
        printf("\n overflow!");
    }
    else if (front==-1 && rear==-1){
        front = rear=0;
    }
    else {
        rear++;
        queue [rear]=num;
    }
}
int delete(){
    int val;
    if (front==-1 || front>rear) {
        print("\n Underflow!");
        retuen -1;
    }
    else {
        val= queue [front];
        front++;
        if (front>rear) {
            front =rear=-1;
        }
        return val;
    }
}
int peek(){
    if (front==-1 || front>rear){
        printf("\n Queue Empty!");
        return-1;
    }
    else {
        return queue[front];
    }
}
```

```
void display() {
  int i;
  if (front== -1 || front > rear) {
    printf ("\n Queue Empty!");
  else {
    for(i= front; i<=rear; i++){
       printf ("\n %d", queue[i]);
    }
  }
}
int main() {
  int choice, value;
  printf ("\n Queue Operation Menu");
  printf ("\n 1. Insert \n2. Delete \n 3. Peek \n 4. Display \n 5. Exit");
  printf( "\n Enter your choice: ");
  scanf ("%d", & ~~optio~~ choice);
  switch (choice){
    case1: insert();
          break;
    case 2: value= delete();
          if (value!=-1){
            printf(" \n Deleted number: %d", value);
          }
          break;
    case 3: value= peek();
          if (value!= -1){
            printf ("\n First value in queue: %d", value);
          }
          break;
    case 4: display();
          break;
    case 5: exit(0);
    default: printf("Input Invalid!");
  }
  return 0;
}
```



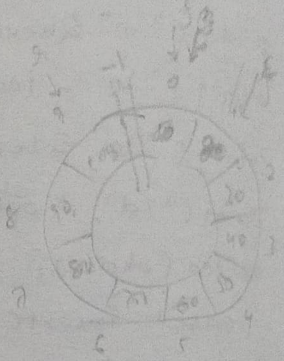
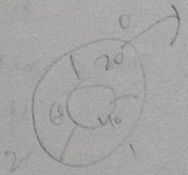
main.c

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

#define MAX_SIZE 3

int queue[MAX_SIZE];
int front = -1, rear = -1;

void insert(int value) {
    if (rear == MAX_SIZE - 1) {
        printf("Queue Overflow: Cannot insert element %d, queue is full\n", value);
    } else {
        if (front == -1) {
            front = 0;
        }
        rear++;
        queue[rear] = value;
        printf("Element %d inserted into the queue\n", value);
    }
}

void delete() {
    if (front == -1) {
        printf("Queue Underflow: Cannot delete element, queue is empty\n");
    } else {
        printf("Element %d deleted from the queue\n", queue[front]);
        if (front == rear) {
            // Reset queue when the last element is deleted
            front = rear = -1;
        } else {
            front++;
        }
    }
```

```
Queue Operations:
1. Insert   2. Delete   3. Display  4. Exit
Enter your choice: 2
Queue Underflow: Cannot delete element, queue is empty

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

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

Queue Operations:
1. Insert   2. Delete   3. Display  4. Exit
Enter your choice: 1
Enter the value to insert: 6
Element 6 inserted into the queue

Queue Operations:
1. Insert   2. Delete   3. Display  4. Exit
Enter your choice: 1
Enter the value to insert: 7
Queue Overflow: Cannot insert element 7, queue is full

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

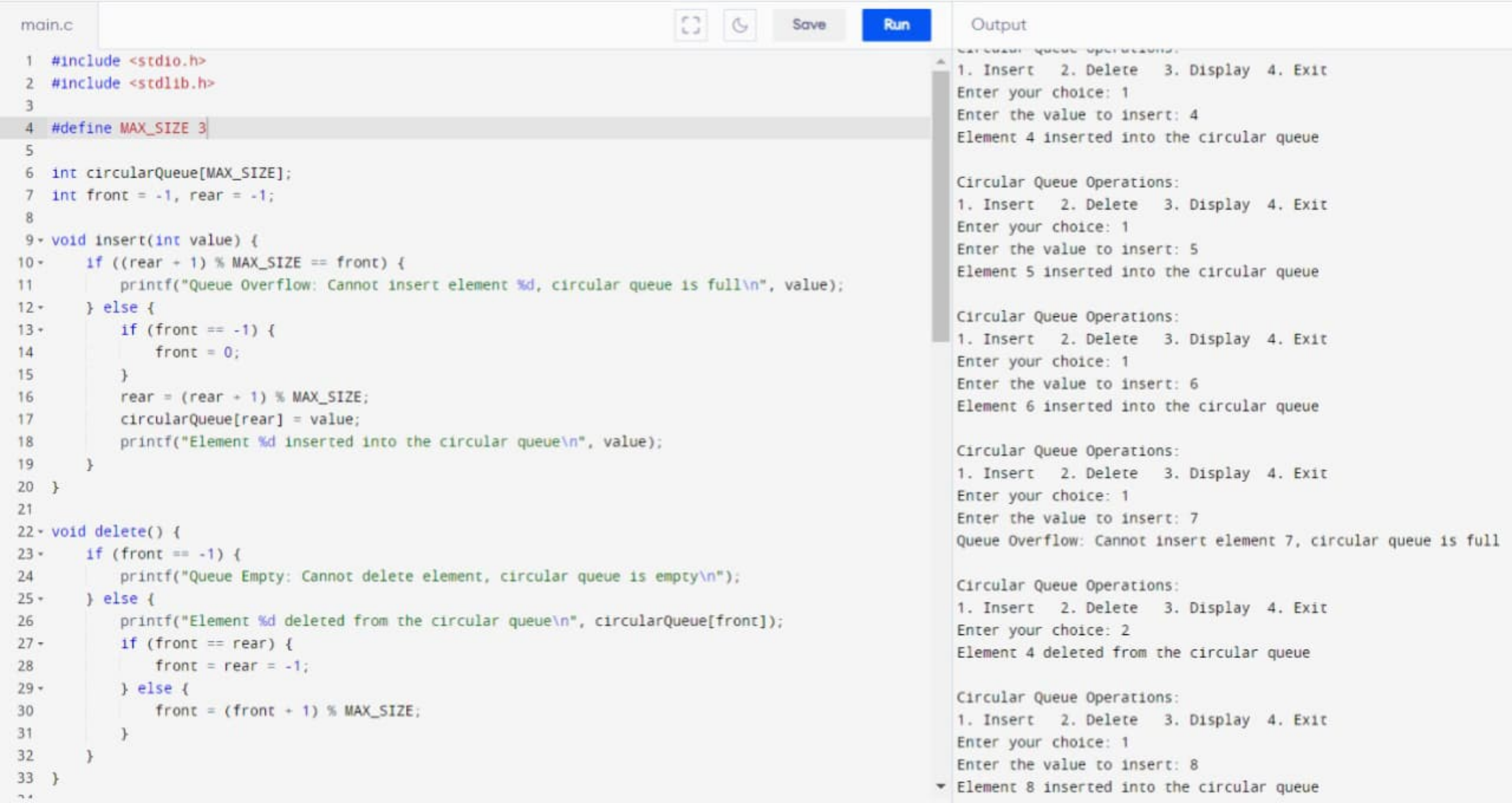
2. CIRCULAR QUEUE USING ARRAYS.

```
# include <stdio.h>
# define MAX [illegible]
int queue [MAX];
int front = -1, rear = -1;
void insert(){
    int num;
    printf("\n Enter number to be inserted: ");
    scanf("%d", &num);
    if (front==0 && rear= MAX-1){
       printf("\n Overflow");
    }
   else if (front==-1 && rear==-1){
      front= rear=0;
      queue [rear] =num;
   }
   else if (rear==MAX-1 && front!=0){
      rear=0;
      queue [rear]=num;
   }
   else {
      rear++;
      queue [rear]=num;
   }
}
~~void~~ int delete(){
 int val;
 if ( front ==-1 && rear==-1){
   printf("\n underflow!);
   return -1;
 }
 val= queue [front];
 if (front==rear) {
   front= rear =-1;
 }
 else {
   if (front == MAX-1) {
     front=0;
   }
   else {
     front ++;
   }
 }
return val;
```

```
void display(){
  int i;
  ~~printf~~if(front==-1 && rear==-1){
     printf("\n Queue Empty!");

  else{
    if(front < rear){
      for(i=front; i<=rear; i++){
         printf("\n %d", queue[i]);
      }
    }
    else{
      for(i=front; i<MAX; i++){
         printf("\n %d", queue[i]);
      }
      for(i=0; i<=rear; i++){
         printf("\n %d") queue[i]);
      }
  }}
}
int main() {
   int choice, value;
   printf("Queue Operation Menu\n");
   printf("\n1. Insert \n 2. Delete \n 3. Display\n 4. Exit);
   printf("Enter a choice: ");
   scanf("%d", &choice);
   switch(choice){
     case1: insert();
            break;
     case 2: val=delete(); ⟶ if (val != -1) printf("\nDeleted number: %d", value);
            break;
     case 3: display();
            break;
     case 4: exit(0);
     default: printf("Input Invalid!");
   }
   return 0;
}
```

[illegible]



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

#define MAX_SIZE 3

int circularQueue[MAX_SIZE];
int front = -1, rear = -1;

void insert(int value) {
    if ((rear + 1) % MAX_SIZE == front) {
        printf("Queue Overflow: Cannot insert element %d, circular queue is full\n", value);
    } else {
        if (front == -1) {
            front = 0;
        }
        rear = (rear + 1) % MAX_SIZE;
        circularQueue[rear] = value;
        printf("Element %d inserted into the circular queue\n", value);
    }
}

void delete() {
    if (front == -1) {
        printf("Queue Empty: Cannot delete element, circular queue is empty\n");
    } else {
        printf("Element %d deleted from the circular queue\n", circularQueue[front]);
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % MAX_SIZE;
        }
    }
}
```

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

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

Circular Queue Operations:
1. Insert   2. Delete   3. Display  4. Exit
Enter your choice: 1
Enter the value to insert: 6
Element 6 inserted into the circular queue

Circular Queue Operations:
1. Insert   2. Delete   3. Display  4. Exit
Enter your choice: 1
Enter the value to insert: 7
Queue Overflow: Cannot insert element 7, circular queue is full

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

Circular Queue Operations:
1. Insert   2. Delete   3. Display  4. Exit
Enter your choice: 1
Enter the value to insert: 8
Element 8 inserted into the circular queue
```

2. <u>Output:</u>

```
Circular Queue operation Menu
1. Insert
2. Delete
3. Display
4. exit
Enter your choice:2
Underflow.
Enter your choice:1
Enter a number:20
Enter your choice:1
Enter a number:40
enter your choice:1
Enter a number:60
Enter your choice:1
Enter a number:80
Overflow.
enter your choice:3
20
40
60
Enter your choice:4
Exit!
```

[illegible]

<u>Output:</u>

```
Linear Queue operation Menu
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice:2
Underflow.
Enter your choice:1
Enter a number:20
Enter your choice:1
Enter a number:40
Enter your choice:1
Enter a number:60
Enter your choice:1
Enter a number:80
Overflow.
Enter your choice:3
20
40
60
```