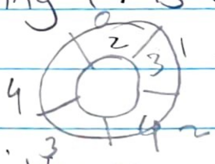
Assignment 3

Aim - Perform functions on a circular queue which follows FIFO (~~Last~~ First in First out). we perform functions to insert, remove and display all elements in the queue

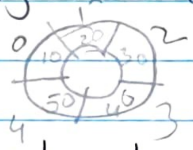
Theory

In a circular queue, there is no end beca
it is circular, we assume our own start and e
positions. To the parent function, this ~~is~~ shou
not matter and the insert and delete functio
should handle the start and end on their own.
It is used ~~as~~ to reclaim unused space in an
array after deletions. It ensures the ~~maxima~~
utilization of the allocated memory to that array.

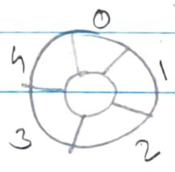
Initialize function - sets the start and end o
the queue to zero, indicating it is empty and
ready to be filled.



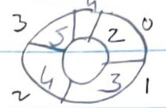
isFull function - If the difference between
start and end index is 1, ie. the queue en
and ~~the~~ starts in the next index of the array
that means it is full ~~otherwise~~



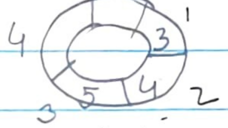
is Empty function - If the array starts and
ends in the same position that means the
queue is full.



enqueue function (Insert) - If the array is full, It prints that it is full or else sets the last element added to the circular array to the value passed to that function and moves the end to the next position, if it is at the end then it comes back to the 0th index.



deque function (Delete) - If the array is empty, it prints that it is empty or else increases the start point to one position higher and takes it to zero if it past the end of the array. It also has a temporary varible value which stores the number removed from the array.



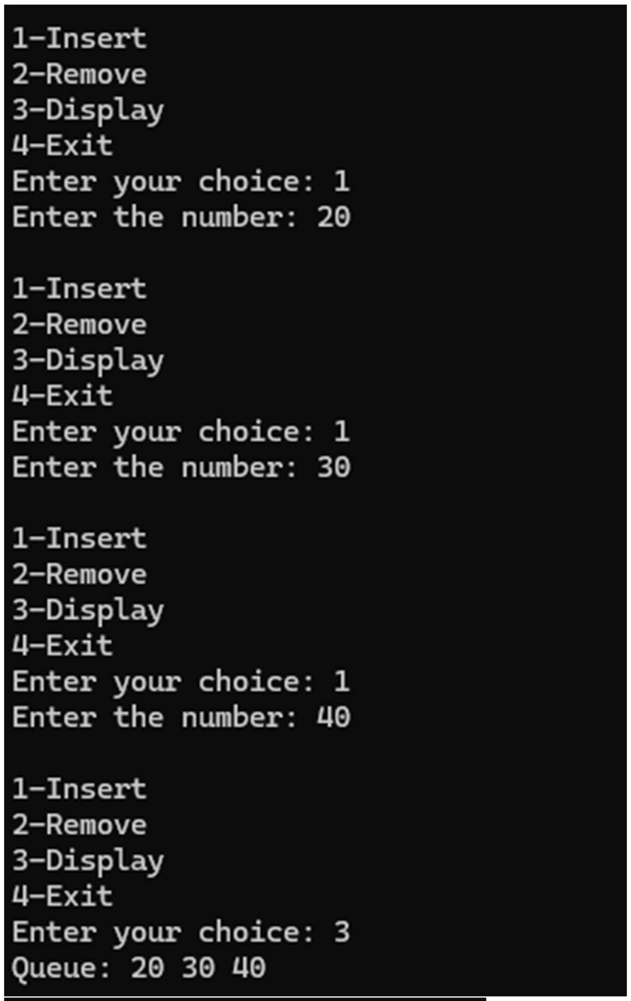
display function - If the queue it loops over all the elements from start to end indexes of the array and prints them.

Time complexity - It is $\Theta(1)$ for all functions except display which is $O(n)$. $O(1)$ means the logic is as efficent as possible and the time required to execute every function is independent of ~~time~~ the number of elements. Only display has time complexity n to iterate overall functions.

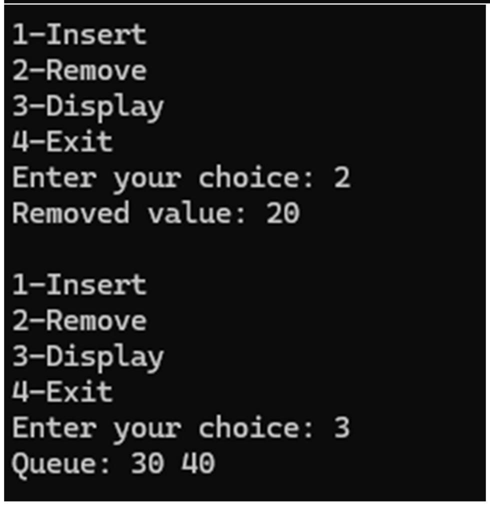
Conclusion
This program implements a cirular queue, following FIFO principle and is optimized for minimum time and resource usage.

```
#include <stdio.h>
#define MAX_SIZE 10

typedef struct {
  int data[MAX_SIZE];
  int start;
  int end;
} Queue;

void initialize(Queue* q) {
  q->start = 0;
  q->end = 0;
}

int isFull(Queue* q) {
  return (q->end + 1) % MAX_SIZE == q->start;
}

int isEmpty(Queue* q) {
  return q->start == q->end;
}

void enqueue(Queue* q, int value) {
  if (isFull(q)) {
    printf("Queue is Full\n");
    return;
  }
  q->data[q->end] = value;
  q->end = (q->end + 1) % MAX_SIZE;
}

int dequeue(Queue* q) {
  if (isEmpty(q)) {
    printf("Queue is Empty\n");
    return -1;
  }
  int value = q->data[q->start];
  q->start = (q->start + 1) % MAX_SIZE;
  return value;
}

void display(Queue* q) {
  if (isEmpty(q)) {
    printf("Queue is Empty\n");
    return;
  }
```

```
    int i = q->start;
    printf("Queue: ");
    while (i != q->end) {
        printf("%d ", q->data[i]);
        i = (i + 1) % MAX_SIZE;
    }
    printf("\n");
}

int main() {
    Queue q;
    initialize(&q);

    int choice, val;

    while (1) {
        printf("\n1-Insert\n2-Remove\n3-Display\n4-Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the number: ");
                scanf("%d", &val);
                enqueue(&q, val);
                break;

            case 2:
                val = dequeue(&q);
                if (val != -1) {
                    printf("Removed value: %d\n", val);
                }
                break;

            case 3:
                display(&q);
                break;

            case 4:
                return 0;

            default:
                printf("Invalid Choice\n");
        }
    }
}
```



```
1-Insert
2-Remove
3-Display
4-Exit
Enter your choice: 1
Enter the number: 20

1-Insert
2-Remove
3-Display
4-Exit
Enter your choice: 1
Enter the number: 30

1-Insert
2-Remove
3-Display
4-Exit
Enter your choice: 1
Enter the number: 40

1-Insert
2-Remove
3-Display
4-Exit
Enter your choice: 3
Queue: 20 30 40
```



```
1-Insert
2-Remove
3-Display
4-Exit
Enter your choice: 2
Removed value: 20

1-Insert
2-Remove
3-Display
4-Exit
Enter your choice: 3
Queue: 30 40
```