Assignment 2b

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

Theory

There is a integer which stores the position of the last element in the queue. when an element is removed, all the elements are moved one space behind. ~~I~~ The rear value is ~~defined~~ initialised as -1.

The ~~is~~ empty function checks if the rear is -1 and returns 1 if it is empty otherwize zero.

The is full function returns 1 if rear is equal to the max size -1 otherwise false.

enqueue adds a number to the end of the queue.

dequeue removes the number at the 0th index and shifts all the following numbers one space behind

A =	10	20	30		

remove 10 (dequeue)

A =	20	30			

display prints the entire queue from the oldest to latest number added.

Time complexity

~~isFull() - O(1) - It is constant as only one~~

enqueue() - O(1) - No shifting, only one function is performed

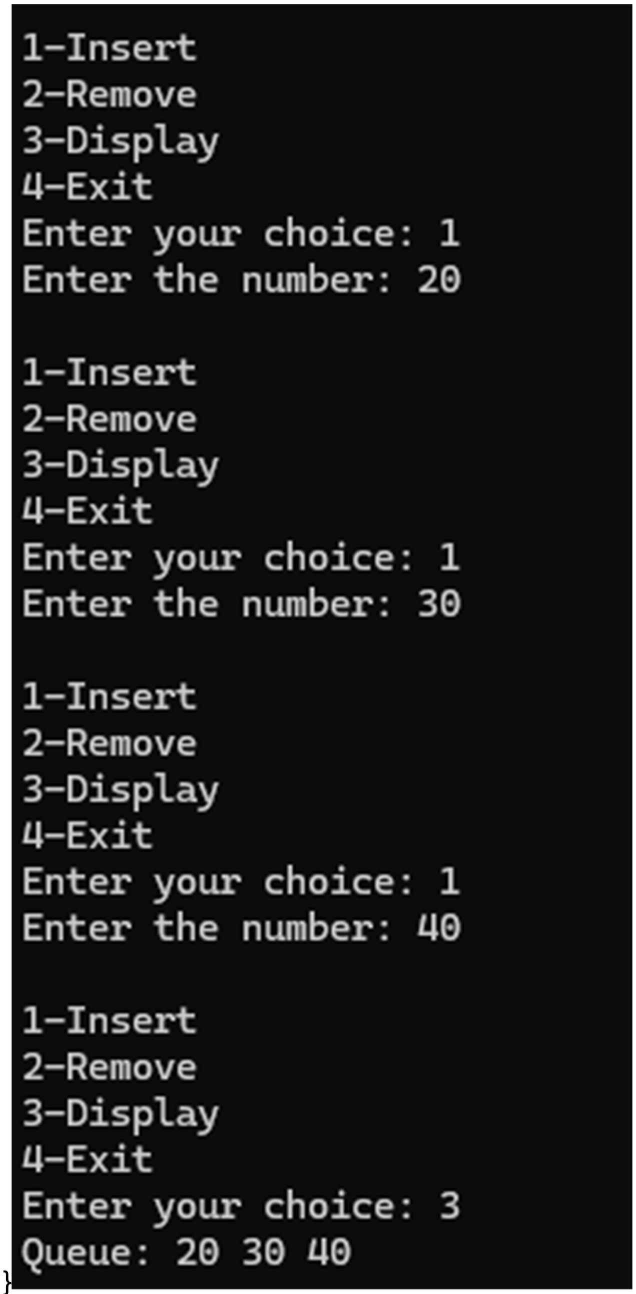
dequeue () - O(n) - All numbers are shifted one space behind

Conclusion

This program successfully implements a circular queue, following FIFO principles. The time complexity ~~shows~~ ~~it is not as~~ is also shown and the individual functions are explained.

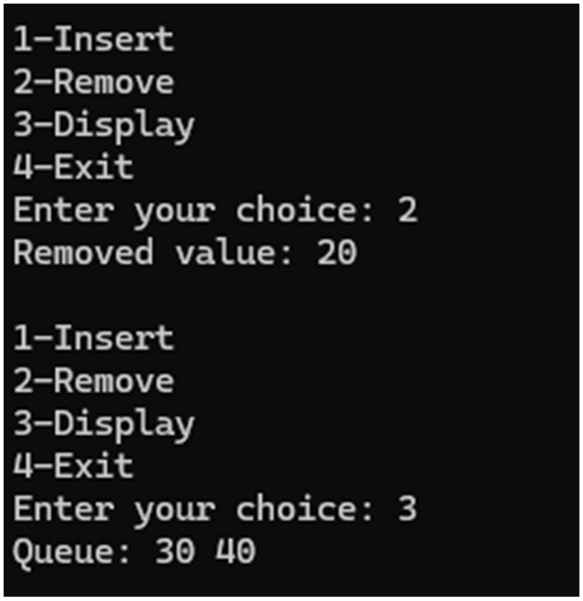
```
#include <stdio.h>
#define MAX_SIZE 10

typedef struct {
   int data[MAX_SIZE];
   int rear;
} Queue;

void initialize(Queue* q) {
   q->rear = -1;
}

int isFull(Queue* q) {
   return q->rear == MAX_SIZE - 1;
}

int isEmpty(Queue* q) {
   return q->rear == -1;
}

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

int dequeue(Queue* q) {
   if (isEmpty(q)) {
      printf("Queue is Empty\n");
      return -1;
   }

   int removed = q->data[0];

   // Left Shift
   for (int i = 0; i < q->rear; i++) {
      q->data[i] = q->data[i + 1];
   }

   q->rear--;
   return removed;
}

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

```
    printf("Queue: ");
    for (int i = 0; i <= q->rear; i++) {
        printf("%d ", q->data[i]);
    }
    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
```