Question:
Design and implement a given type of (ordinary queue, circular queue) queue in C (array implementation/ Linked list implementation). And demonstrate its working with suitable
inputs. Display appropriate messages in case of exceptions.

Aim:
To Implement Circular Queue using Linked Lists

Algorithm:

Enqueue:

- Create a new node.
- If the value of the node is NULL that means the system has run out of memory to allocate so throw an exception indicating Overflow and return the control flow
- Else make the value of the node the data that the user provided
- Also if ‘front’ and ‘rear’ point to NULL then make them point towards the newly created node.
- Also make rear’s next element point to front
- If ‘front’ doesn't point to NULL then make rear’s next value point to the newly created node
- Now make rear point to the new node
- And set rear’s next to front

Dequeue:

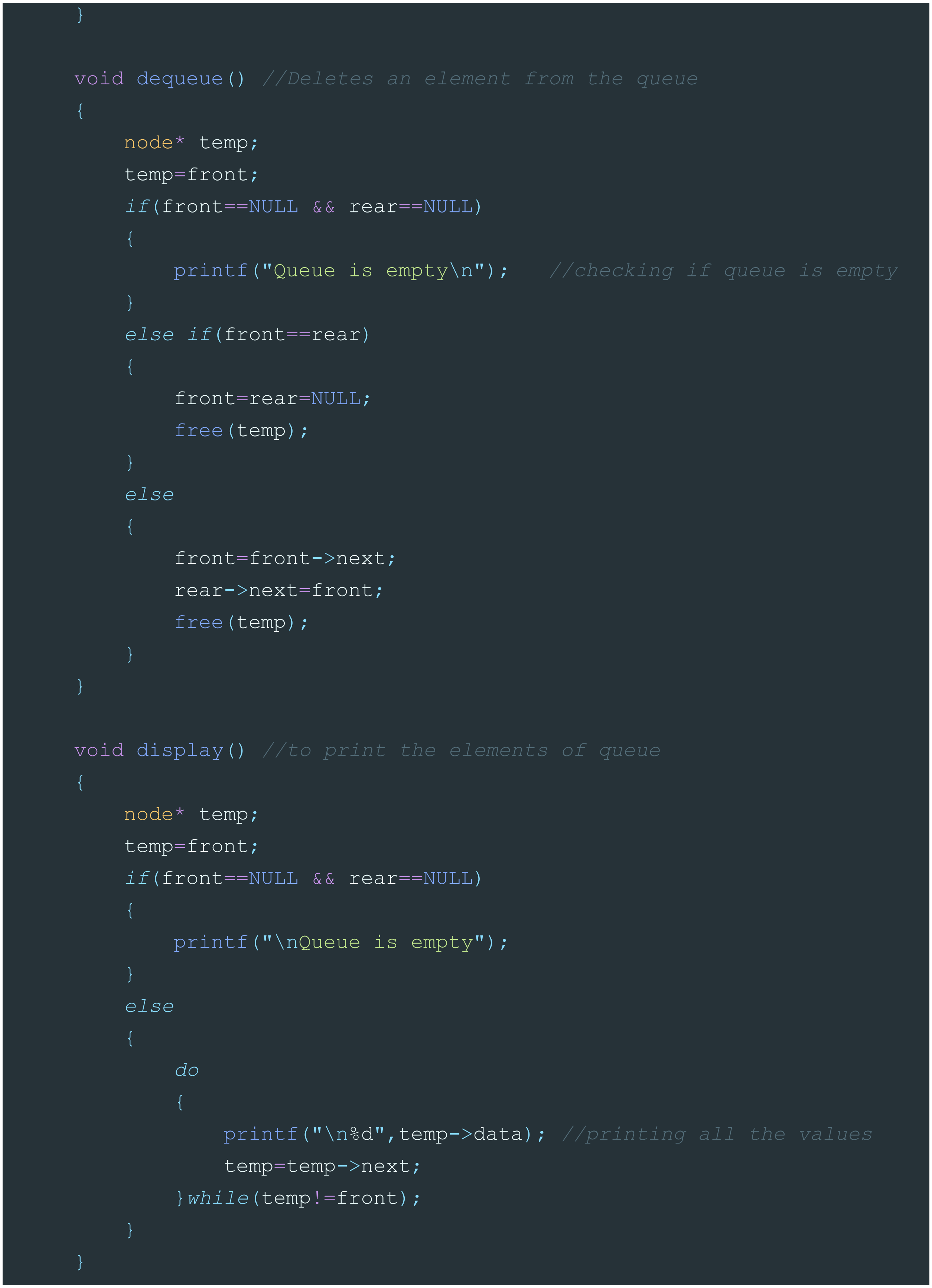
- Firstly check if front is NULL
- If it is NULL then it means that the queue is empty and we run into the underflow situation, so throw an exception indicating underflow and return the control flow
- Now check if front and rear point to the same node (to check if there’s only one element in queue)
- If they do, then set them both to NULL
- If none of the above conditions are satisfied
- Then make front it’s next element
- And set rear’s next element to front

Display

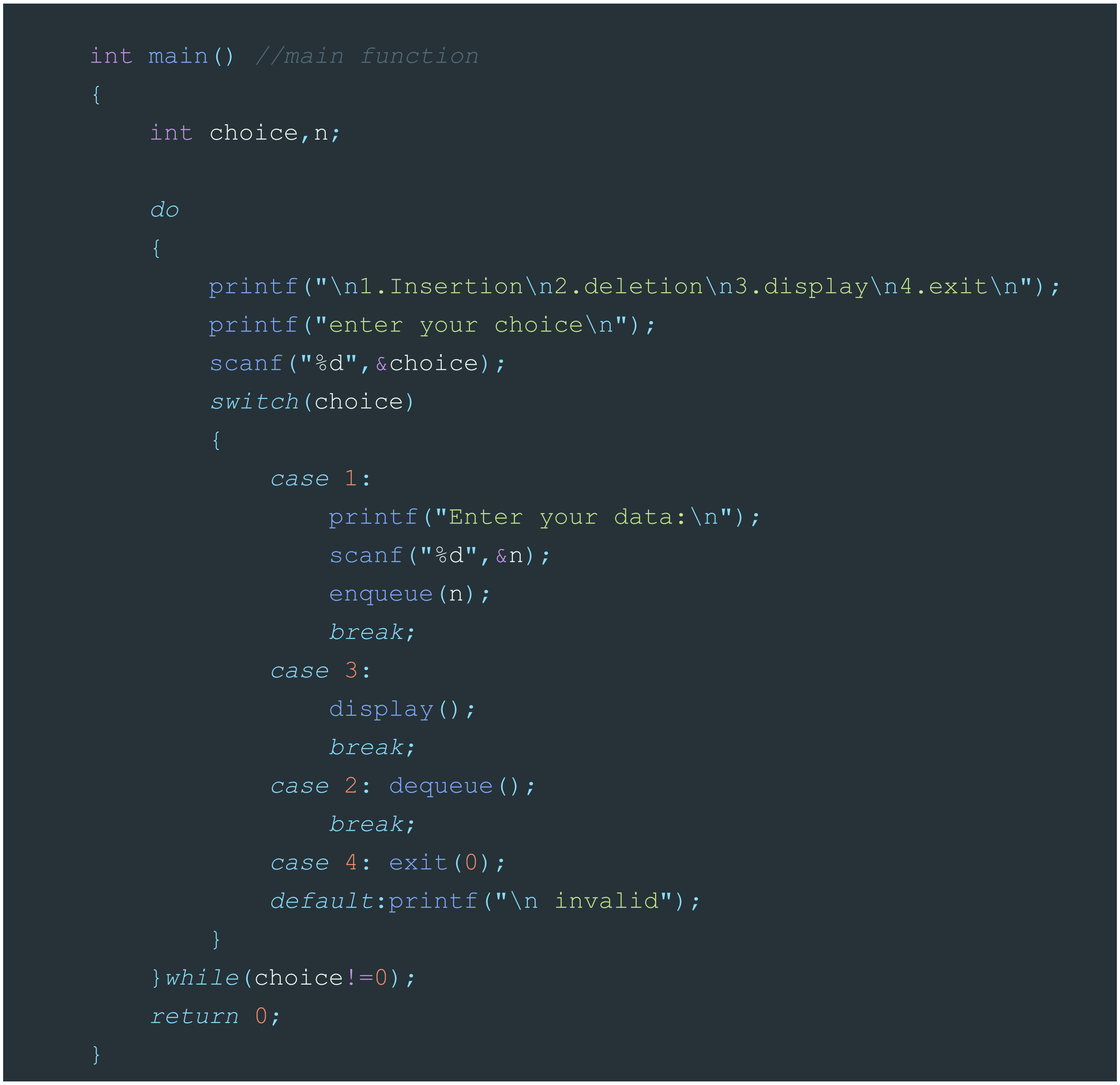
- Make a new node pointer which points to ‘front’.
- If the pointer points to null then display a message saying that the queue is empty
- Or else print the value of the current and then make the pointer point towards the next node
- Repeat step 3 till the pointer points back to front.

Program

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
typedef struct node node;
node *front=NULL;
node *rear=NULL;
void enqueue(int x){ //To insert elements into the circular queue
    node* ptr;
    ptr=(node*)malloc(sizeof(node));
    if(ptr==NULL){
        printf("ran out of memory");
        return;
    }
    ptr->data=x;
    ptr->next=NULL;//0
    if(rear==NULL && front==NULL)
    {
        front=rear=ptr;
        rear->next=front;
    }
    else
    {
        rear->next=ptr;
        rear=ptr;
        rear->next=front;
    }
```

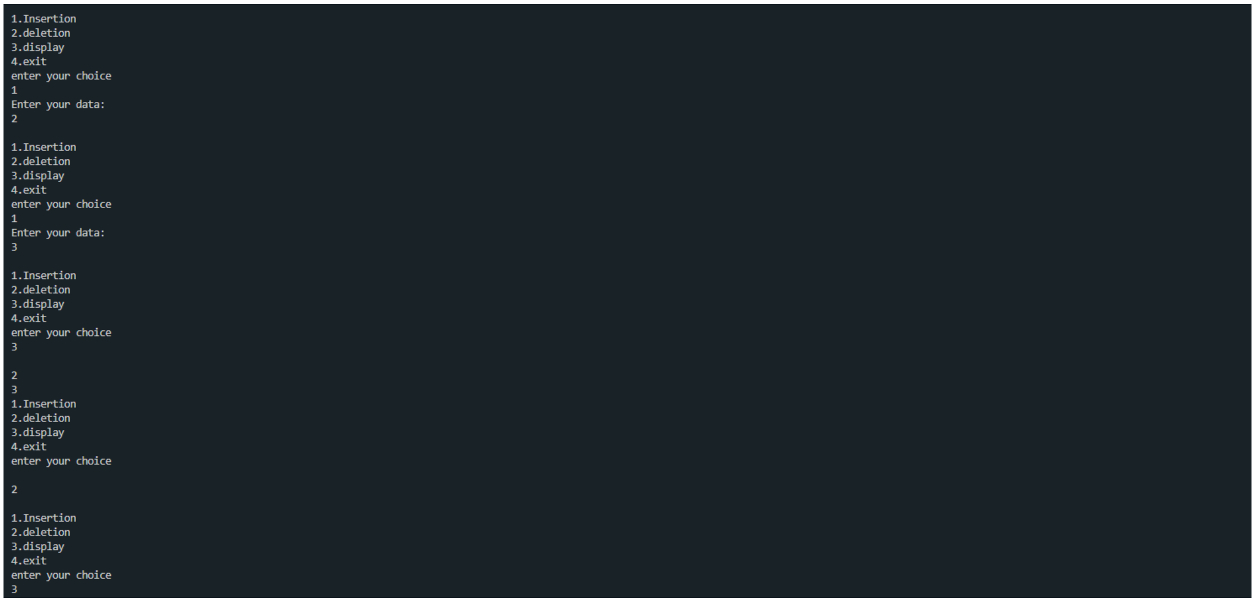


```
}

void dequeue() //Deletes an element from the queue
{
    node* temp;
    temp=front;
    if(front==NULL && rear==NULL)
    {
        printf("Queue is empty\n");   //checking if queue is empty
    }
    else if(front==rear)
    {
        front=rear=NULL;
        free(temp);
    }
    else
    {
        front=front->next;
        rear->next=front;
        free(temp);
    }
}

void display() //to print the elements of queue
{
    node* temp;
    temp=front;
    if(front==NULL && rear==NULL)
    {
        printf("\nQueue is empty");
    }
    else
    {
        do
        {
            printf("\n%d",temp->data); //printing all the values
            temp=temp->next;
        }while(temp!=front);
    }
}
```



```c
int main() //main function
{
    int choice,n;

    do
    {
        printf("\n1.Insertion\n2.deletion\n3.display\n4.exit\n");
        printf("enter your choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("Enter your data:\n");
                scanf("%d",&n);
                enqueue(n);
                break;
            case 3:
                display();
                break;
            case 2: dequeue();
                break;
            case 4: exit(0);
            default:printf("\n invalid");
        }
    }while(choice!=0);
    return 0;
}
```

Output



```
1.Insertion
2.deletion
3.display
4.exit
enter your choice
1
Enter your data:
2

1.Insertion
2.deletion
3.display
4.exit
enter your choice
1
Enter your data:
3

1.Insertion
2.deletion
3.display
4.exit
enter your choice
3

2
3
1.Insertion
2.deletion
3.display
4.exit
enter your choice

2

1.Insertion
2.deletion
3.display
4.exit
enter your choice
3
```