12/1/22

Reverse a_Queue using Stack.

Aim

Write a program to reverse content of que using. Stack.

Algorithm

```
step 1: Start
step 2: Define a structure node that contains.
          (1) Int data.
          (2) pointer to structure node, link.
step 3: Declare 3 variables of node *front, *rear, *top.
step 4: U~~sing infinite loo~~p Inside function.
        enque that accepts an integer value.
          1) Allocate memory for struct node*temp.
          2) temp->data=value
          3 temp-> link=NULL
          4 if front==NUL
                  front=rear=temp.

          5 else
               rear->link =temp
               rear=temp
```

step 5) Inside the function deque.

1) if front=NULL display the queue is empty

2) else temp=front.
allocate memory for newnode
newnode->data= temp->data
newnode->link= top.
top= newnode.
front= front->link.
free the memory of temp.

3)

step 6) Inside function display

① if front== NULL.
display the queue is empty

② else display the nodes from front until the node is NULL.

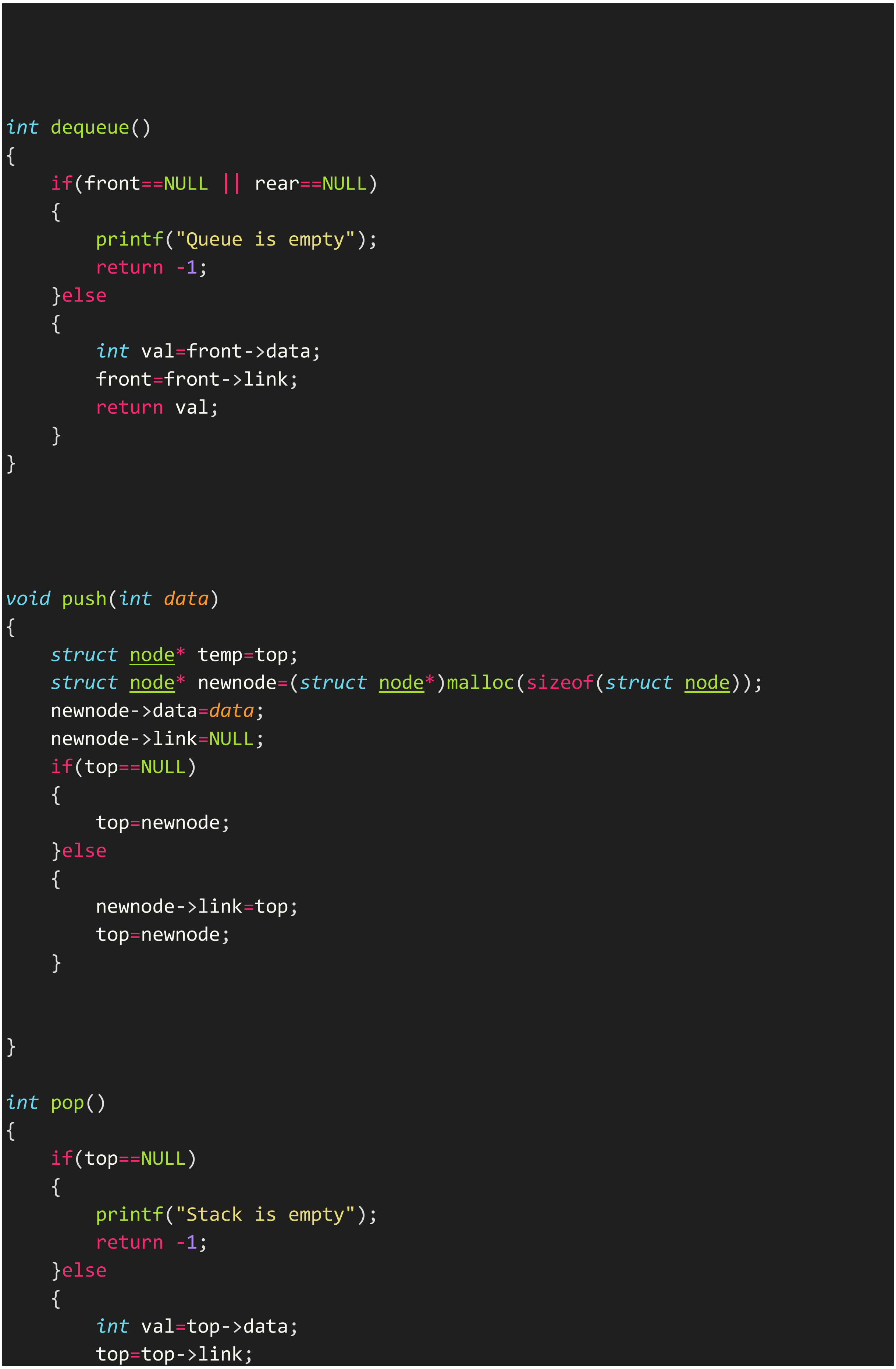
step 7: Inside the function reverse.

① ~~temp= top~~.
② ~~while temp != NUL~~
~~pop the eleme deque~~ th.
① deque all the elements from the queue. and push it to Stack.
②- pop all elements from stack and ~~push~~ enque it
③ display the stack

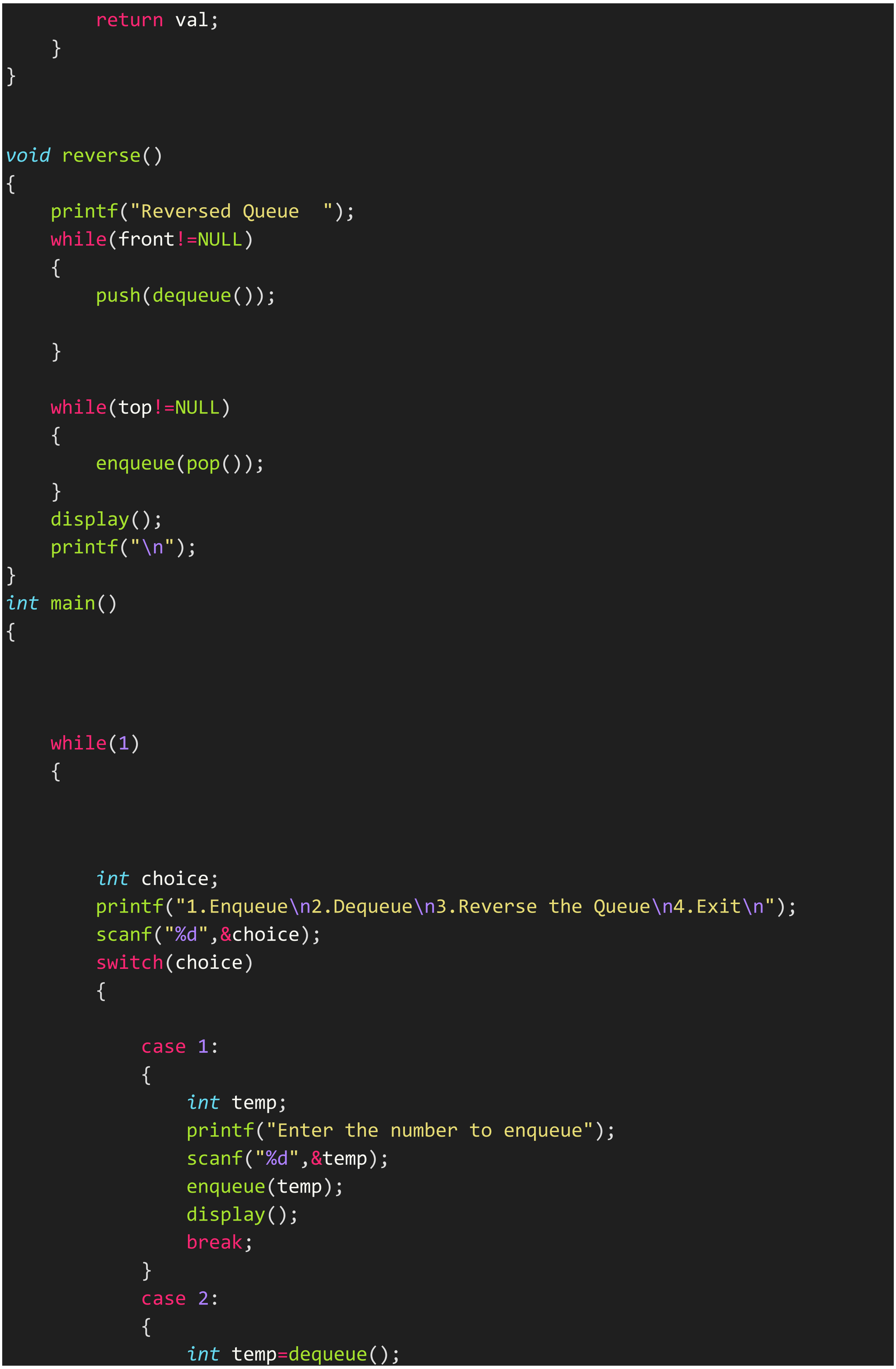
step 8: Stop.

Code

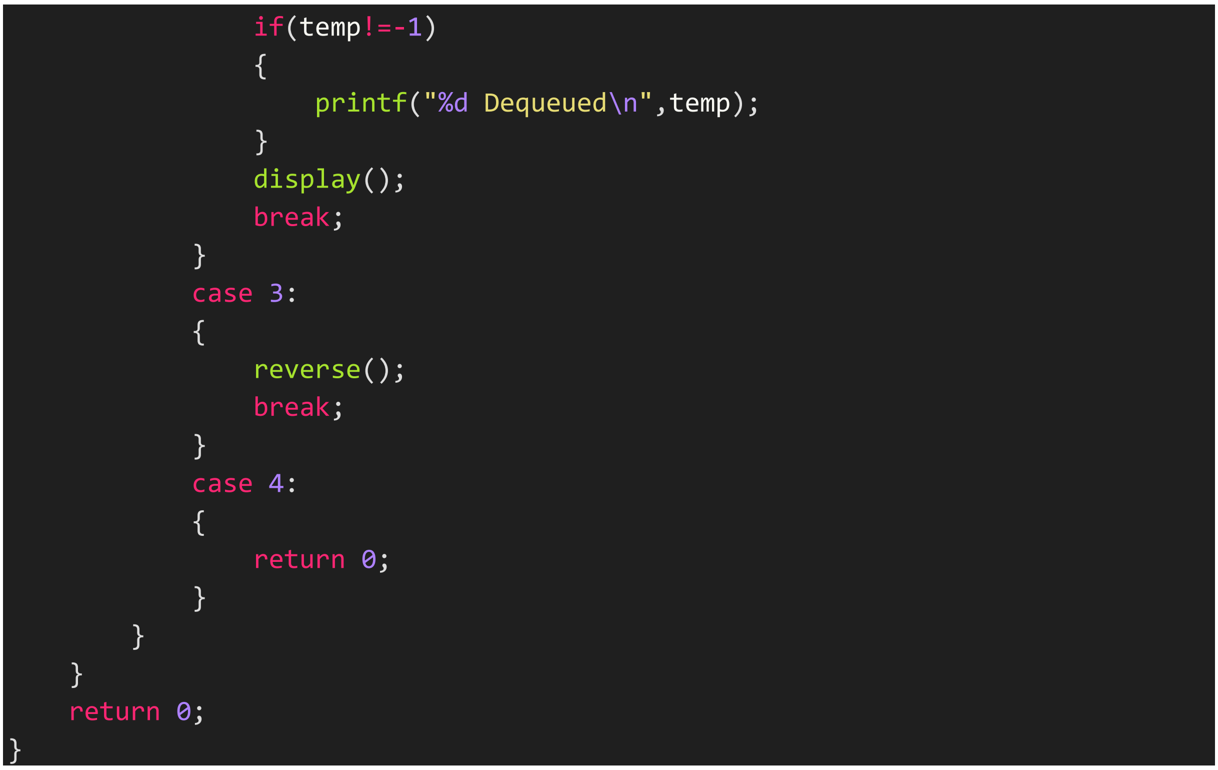
```
#include<stdio.h>
#include<stdlib.h>


struct node
{
    int data;
    struct node* link;
};

struct node* top=NULL;
struct node* front=NULL;
struct node* rear=NULL;




void display()
{
    struct node* temp=front;
    while(temp!=NULL)
    {
        printf("%d ",temp->data);
        temp=temp->link;
    }
    printf("\n");
}




void enqueue(int data)
{
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->link=NULL;
    if(front==NULL || rear==NULL)
    {
        front=rear=newnode;
    }else
    {
        rear->link=newnode;
        rear=newnode;
    }
}
```



```c
int dequeue()
{
    if(front==NULL || rear==NULL)
    {
        printf("Queue is empty");
        return -1;
    }else
    {
        int val=front->data;
        front=front->link;
        return val;
    }
}



void push(int data)
{
    struct node* temp=top;
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->link=NULL;
    if(top==NULL)
    {
        top=newnode;
    }else
    {
        newnode->link=top;
        top=newnode;
    }


}

int pop()
{
    if(top==NULL)
    {
        printf("Stack is empty");
        return -1;
    }else
    {
        int val=top->data;
        top=top->link;
```



```c
        return val;
    }
}


void reverse()
{
    printf("Reversed Queue  ");
    while(front!=NULL)
    {
        push(dequeue());

    }

    while(top!=NULL)
    {
        enqueue(pop());
    }
    display();
    printf("\n");
}
int main()
{



    while(1)
    {



        int choice;
        printf("1.Enqueue\n2.Dequeue\n3.Reverse the Queue\n4.Exit\n");
        scanf("%d",&choice);
        switch(choice)
        {

            case 1:
            {
                int temp;
                printf("Enter the number to enqueue");
                scanf("%d",&temp);
                enqueue(temp);
                display();
                break;
            }
            case 2:
            {
                int temp=dequeue();
```



```
                if(temp!=-1)
                {
                    printf("%d Dequeued\n",temp);
                }
                display();
                break;
            }
            case 3:
            {
                reverse();
                break;
            }
            case 4:
            {
                return 0;
            }
        }
    }
    return 0;
}
```

OUTPUT

```
1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue12

12

1.Enqueue

2.Dequeue

3.Reverse the Queue

4.Exit

1

Enter the number to enqueue13

12 13

1.Enqueue

2.Dequeue
```

```
3.Reverse the Queue
4.Exit
1
Enter the number to enqueue14
12 13 14
1.Enqueue
2.Dequeue
3.Reverse the Queue
4.Exit
1
Enter the number to enqueue15
12 13 14 15
1.Enqueue
2.Dequeue
3.Reverse the Queue
4.Exit
1
Enter the number to enqueue16
12 13 14 15 16
1.Enqueue
2.Dequeue
3.Reverse the Queue
4.Exit
2
12 Dequeued
13 14 15 16
1.Enqueue
2.Dequeue
3.Reverse the Queue
4.Exit
3
Reversed Queue  16 15 14 13
```

```
1.Enqueue
2.Dequeue
3.Reverse the Queue
4.Exit
4
```