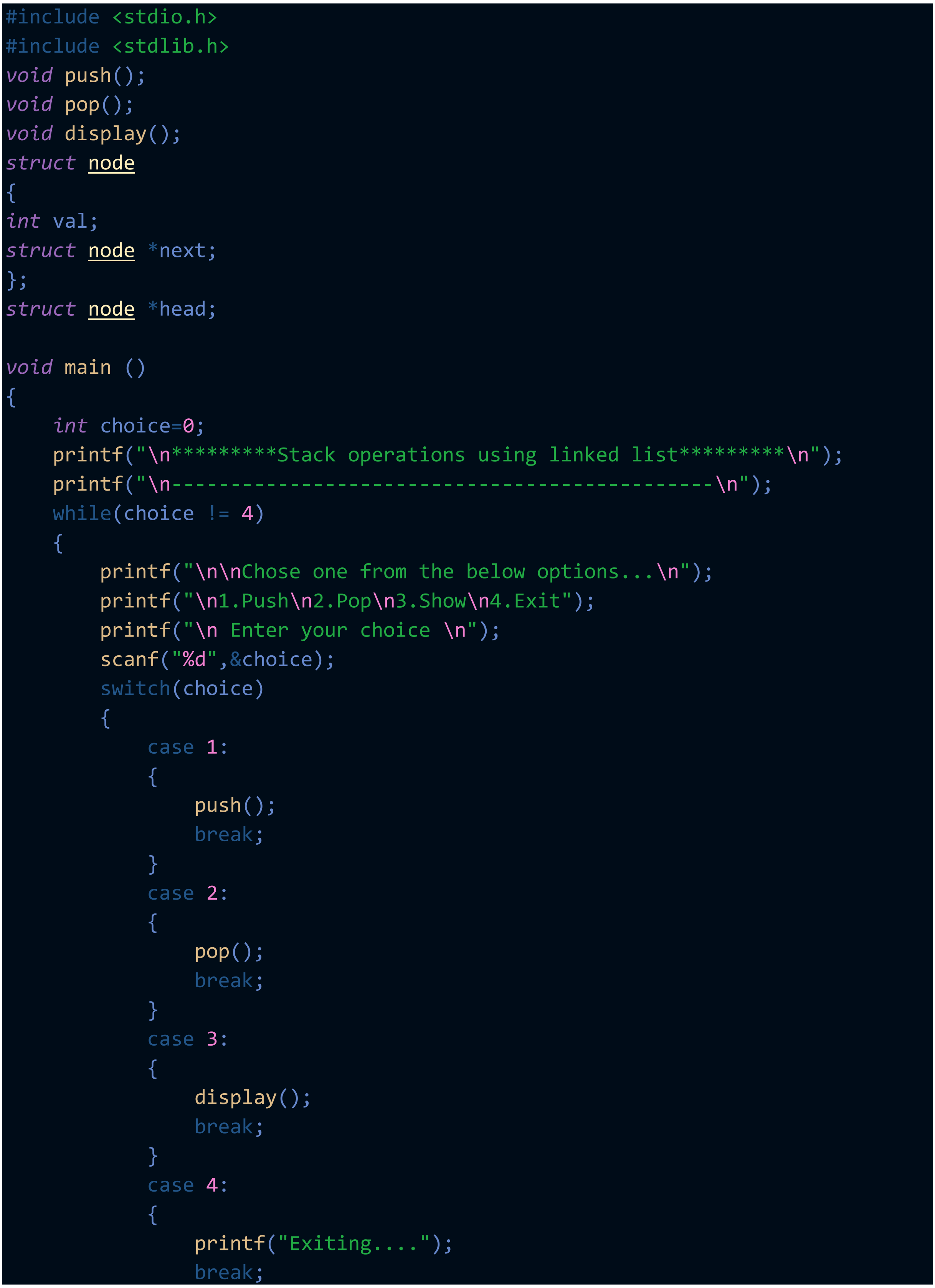
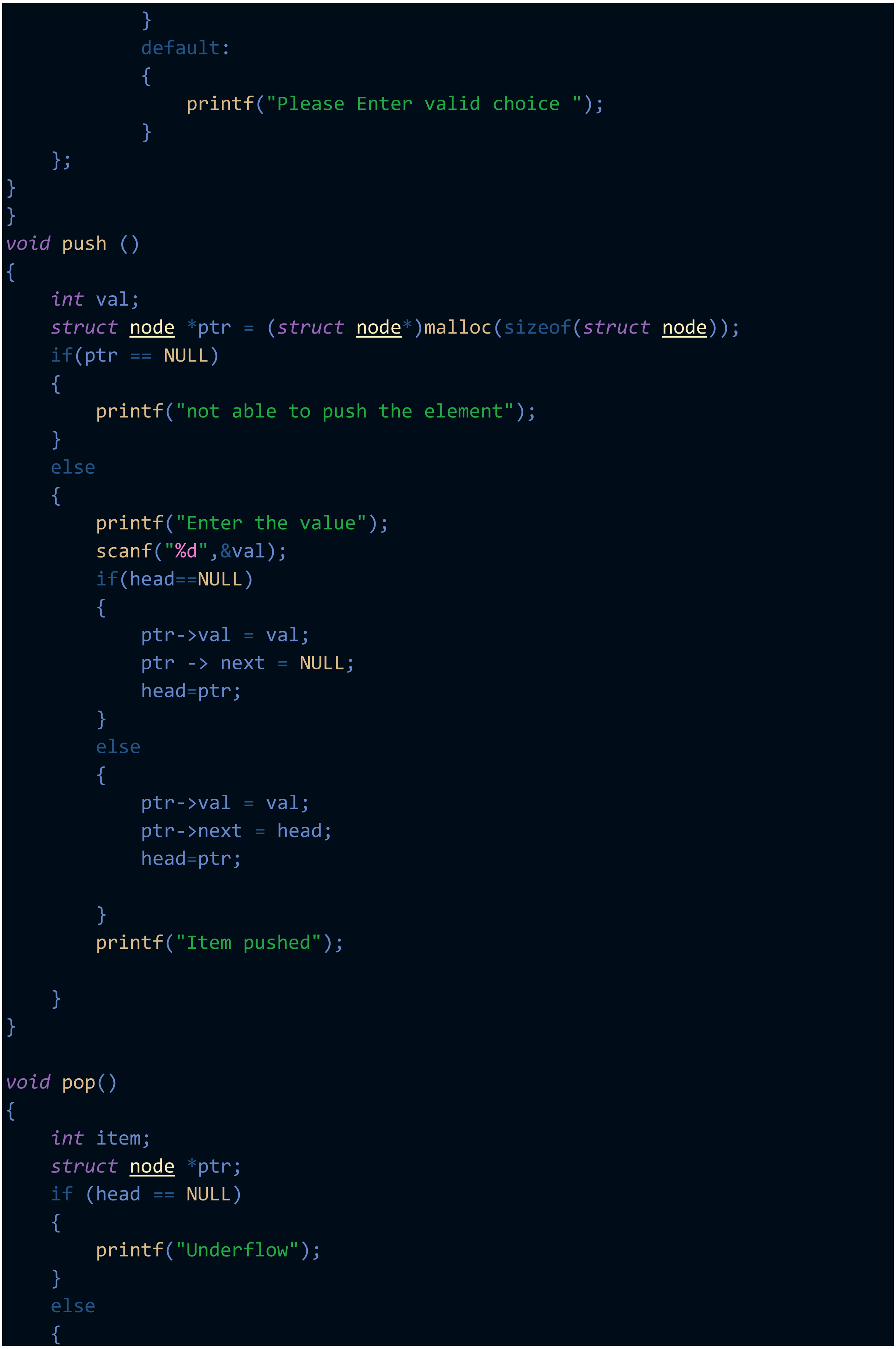
Assignment No: 2

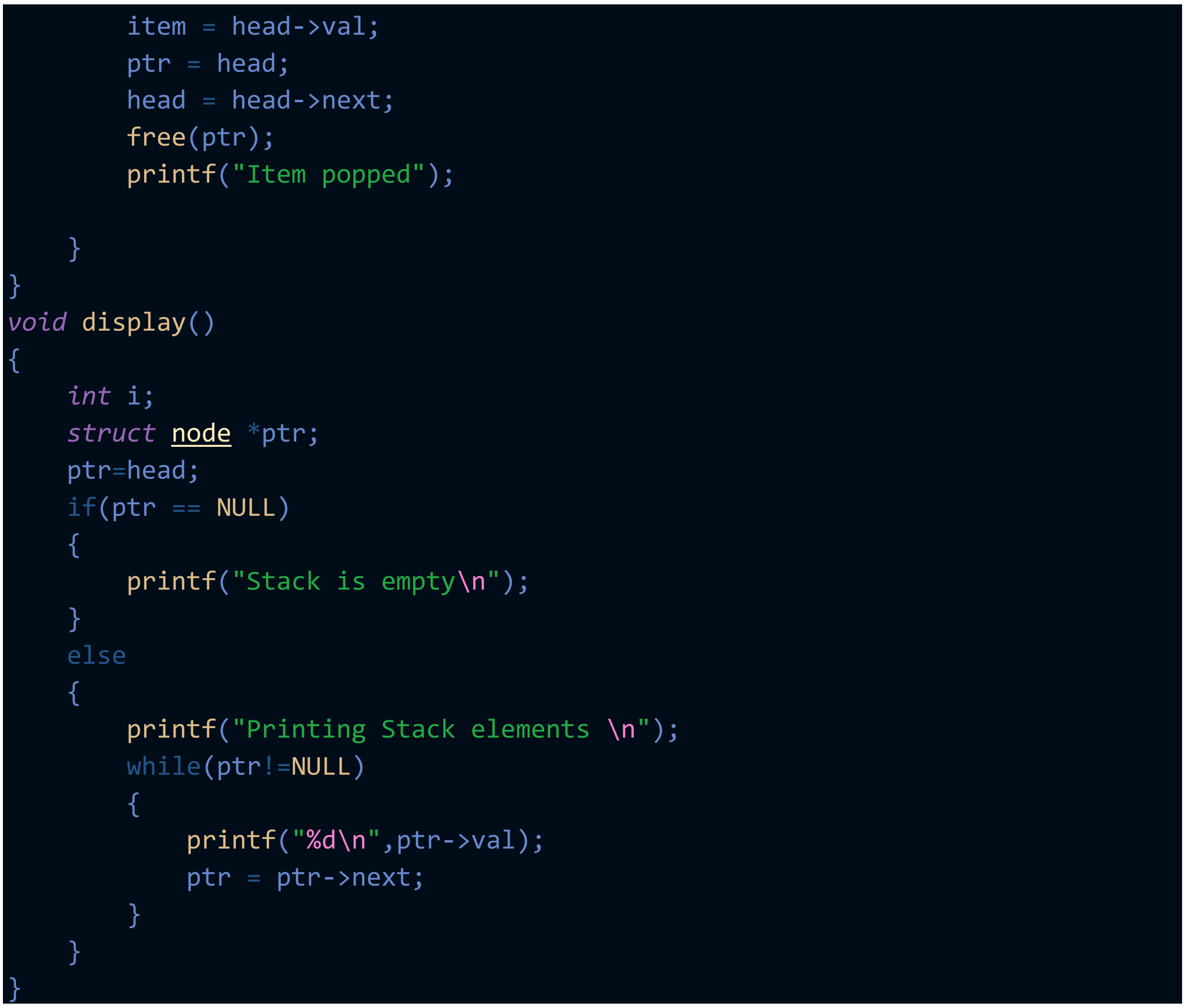
Stack using Linkedlist:



```c
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

void main ()
{
    int choice=0;
    printf("\n*********Stack operations using linked list*********\n");
    printf("\n----------------------------------------------\n");
    while(choice != 4)
    {
        printf("\n\nChose one from the below options...\n");
        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
        printf("\n Enter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("Exiting....");
                break;
```

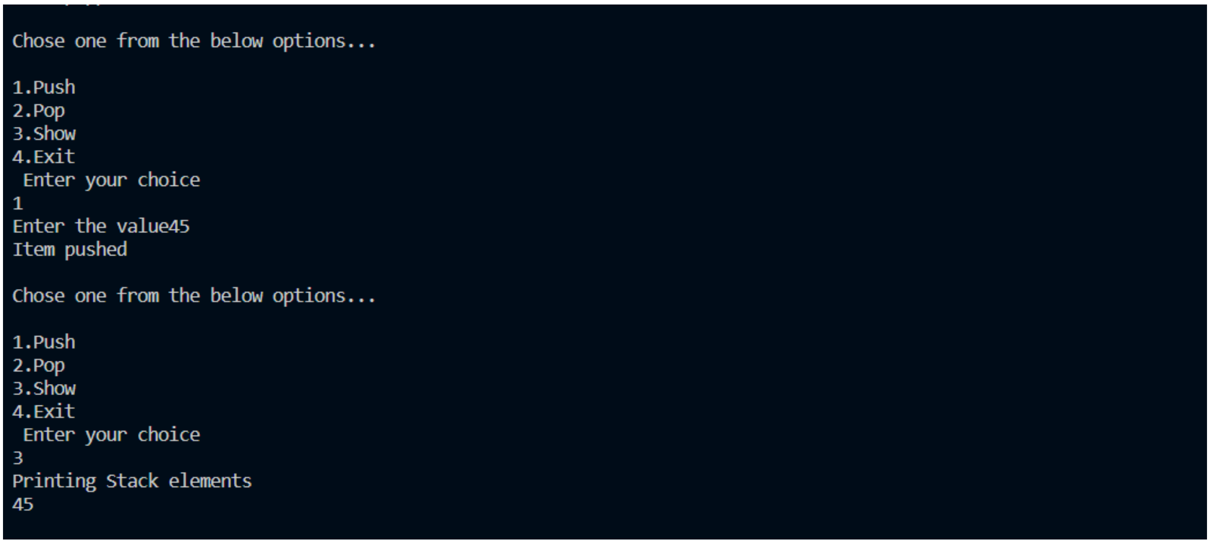


```
            }
            default:
            {
                printf("Please Enter valid choice ");
            }
    };
}
}
void push ()
{
    int val;
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("not able to push the element");
    }
    else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(head==NULL)
        {
            ptr->val = val;
            ptr -> next = NULL;
            head=ptr;
        }
        else
        {
            ptr->val = val;
            ptr->next = head;
            head=ptr;

        }
        printf("Item pushed");

    }
}

void pop()
{
    int item;
    struct node *ptr;
    if (head == NULL)
    {
        printf("Underflow");
    }
    else
    {
```



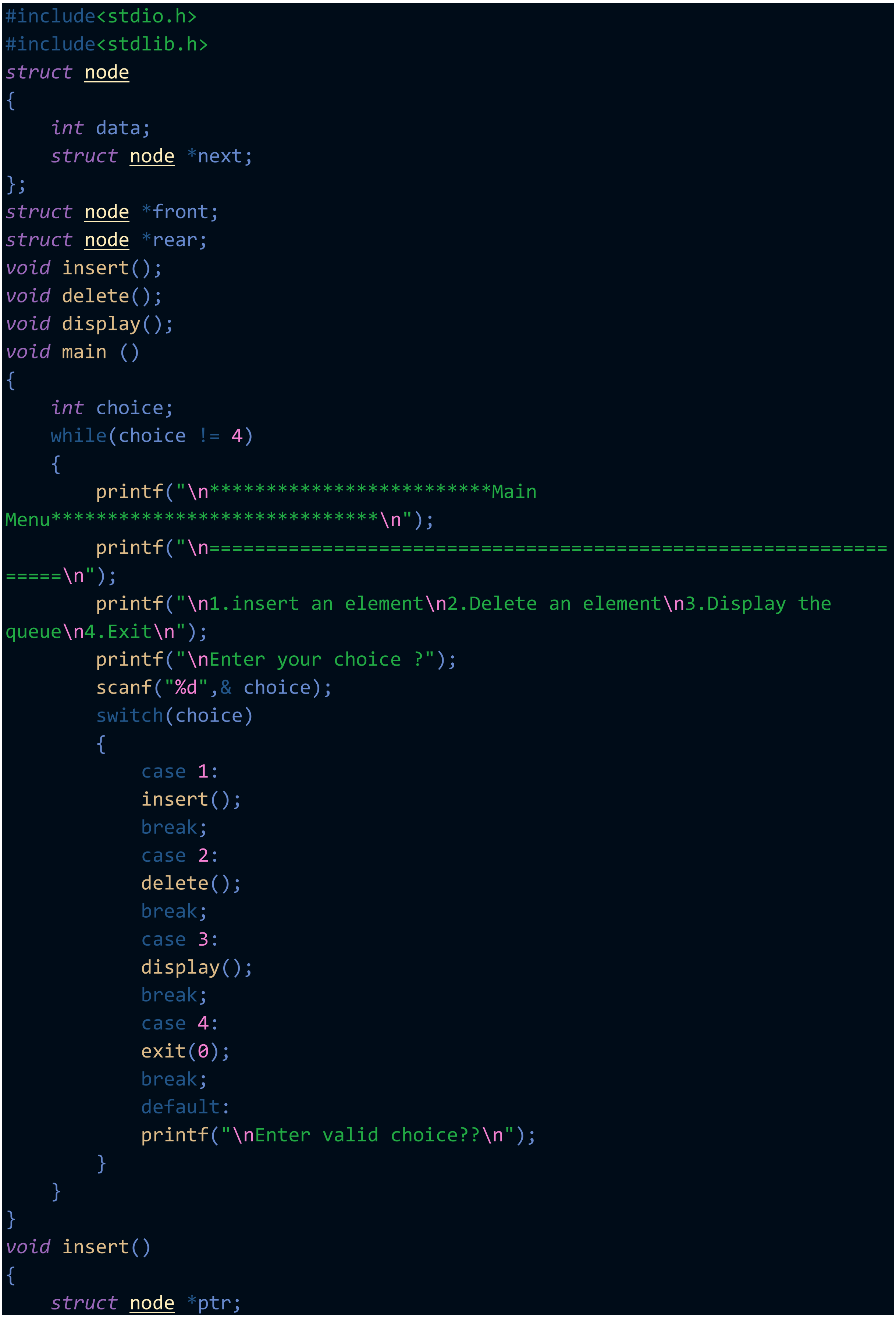
```c
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");

    }
}
void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
```

Output:

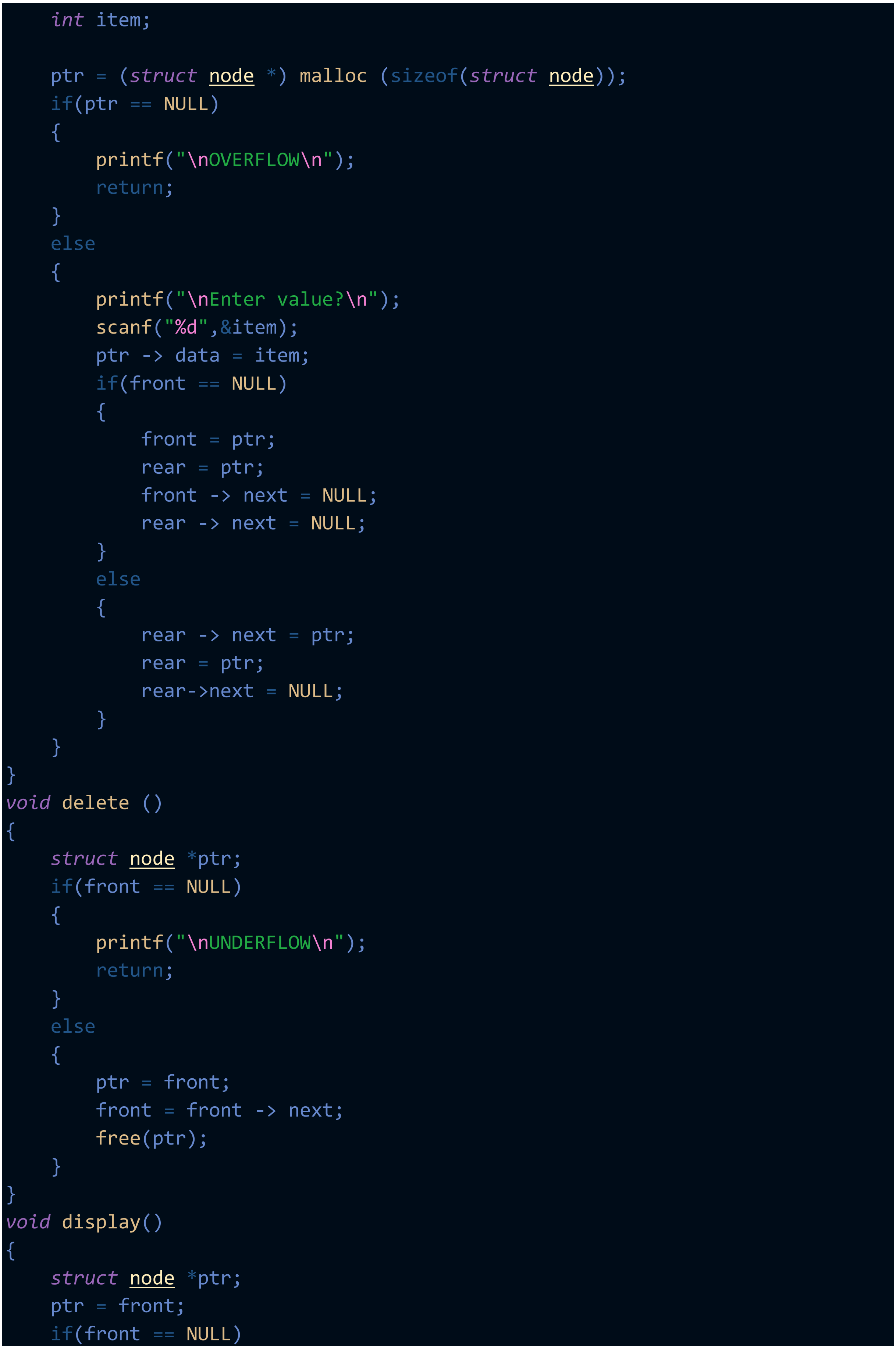


```
Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
1
Enter the value45
Item pushed

Chose one from the below options...

1.Push
2.Pop
3.Show
4.Exit
 Enter your choice
3
Printing Stack elements
45
```

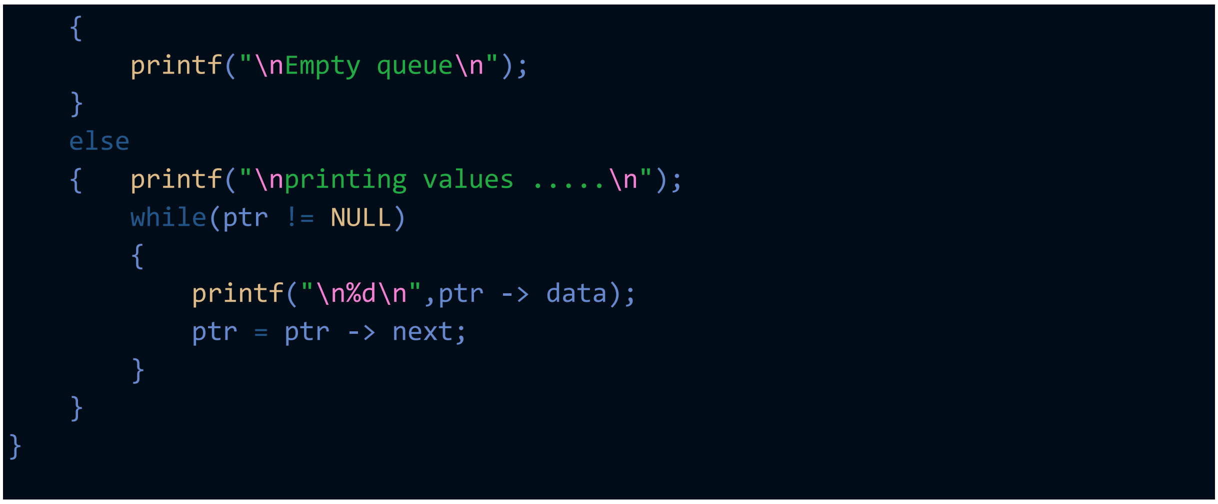
Queue using Linkedlist:



```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
    int choice;
    while(choice != 4)
    {
        printf("\n*************************Main
Menu*****************************\n");
        printf("\n============================================================
=====\n");
        printf("\n1.insert an element\n2.Delete an element\n3.Display the
queue\n4.Exit\n");
        printf("\nEnter your choice ?");
        scanf("%d",& choice);
        switch(choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(0);
            break;
            default:
            printf("\nEnter valid choice??\n");
        }
    }
}
void insert()
{
    struct node *ptr;
```

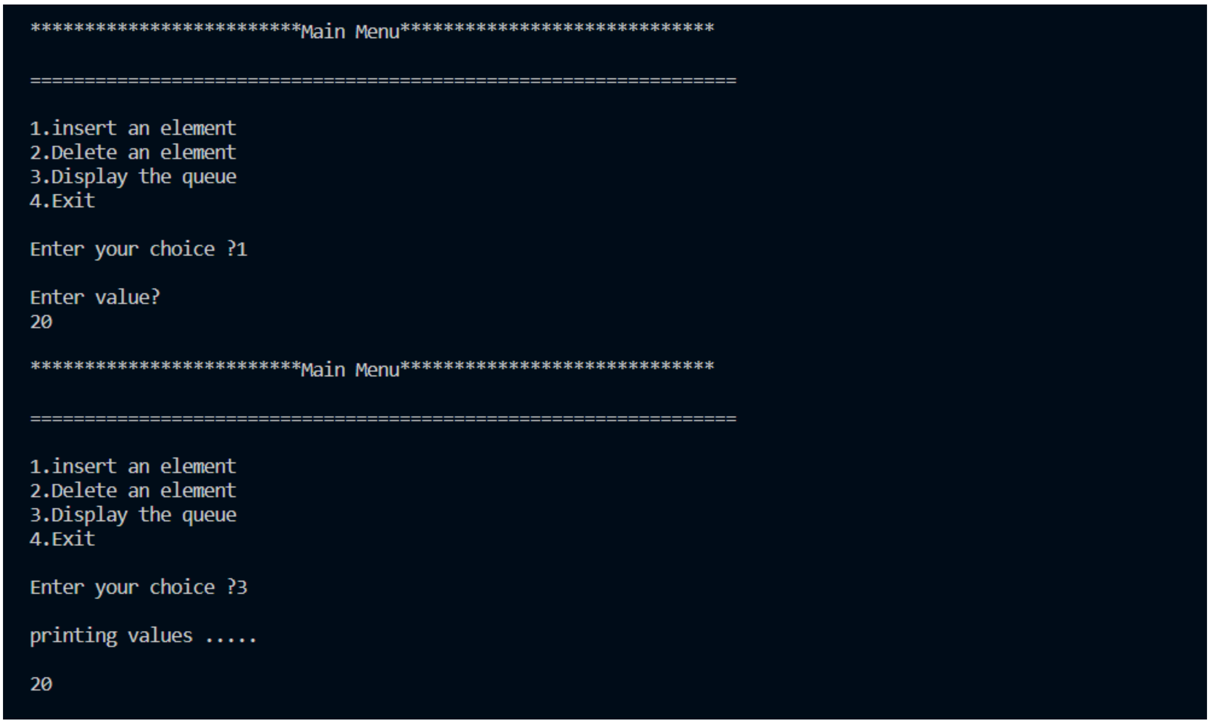


```
    int item;

    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
        return;
    }
    else
    {
        printf("\nEnter value?\n");
        scanf("%d",&item);
        ptr -> data = item;
        if(front == NULL)
        {
            front = ptr;
            rear = ptr;
            front -> next = NULL;
            rear -> next = NULL;
        }
        else
        {
            rear -> next = ptr;
            rear = ptr;
            rear->next = NULL;
        }
    }
}
void delete ()
{
    struct node *ptr;
    if(front == NULL)
    {
        printf("\nUNDERFLOW\n");
        return;
    }
    else
    {
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}
void display()
{
    struct node *ptr;
    ptr = front;
    if(front == NULL)
```



```
    {
        printf("\nEmpty queue\n");
    }
    else
    {   printf("\nprinting values .....\n");
        while(ptr != NULL)
        {
            printf("\n%d\n",ptr -> data);
            ptr = ptr -> next;
        }
    }
}
```

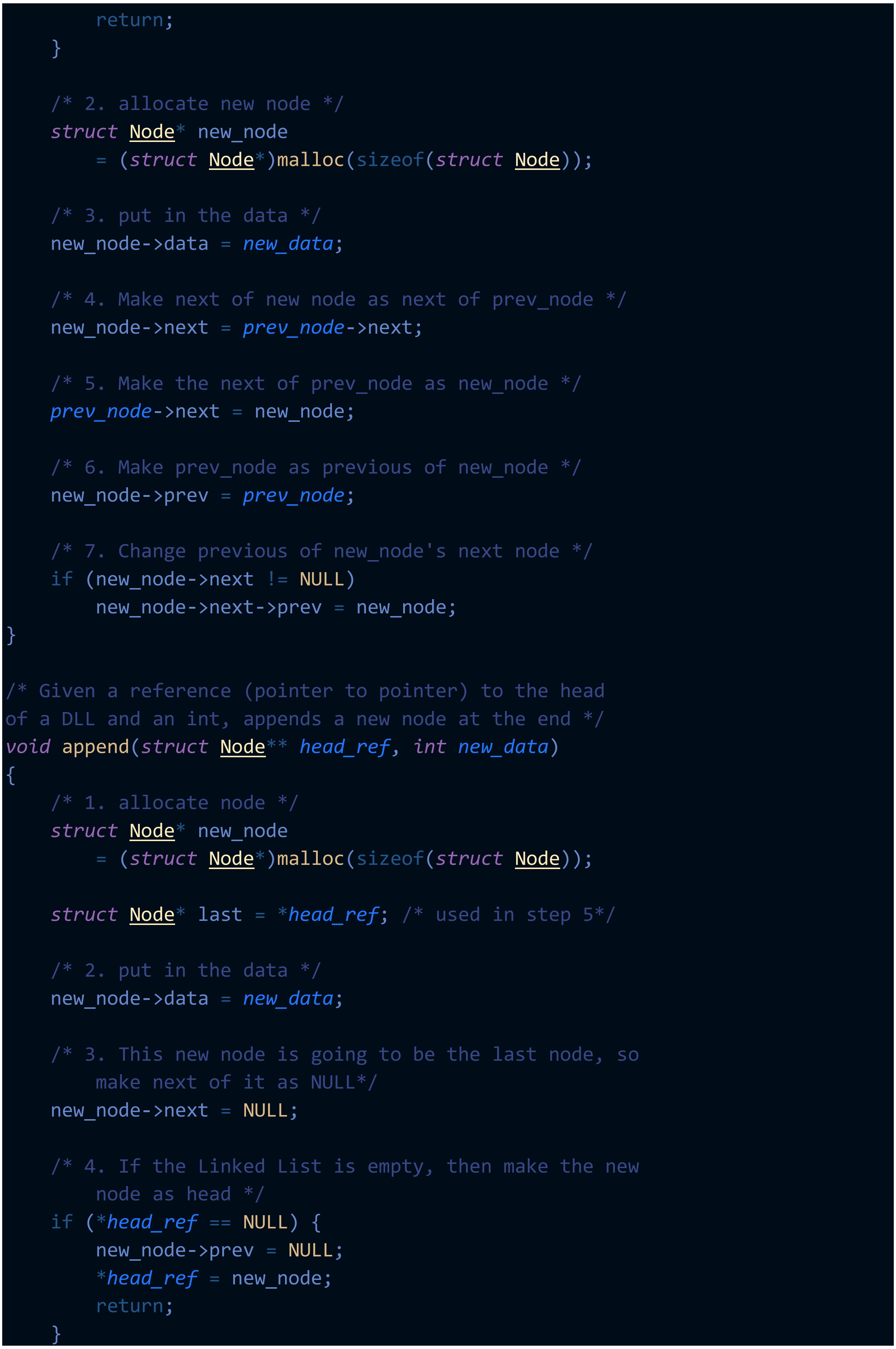
Output:



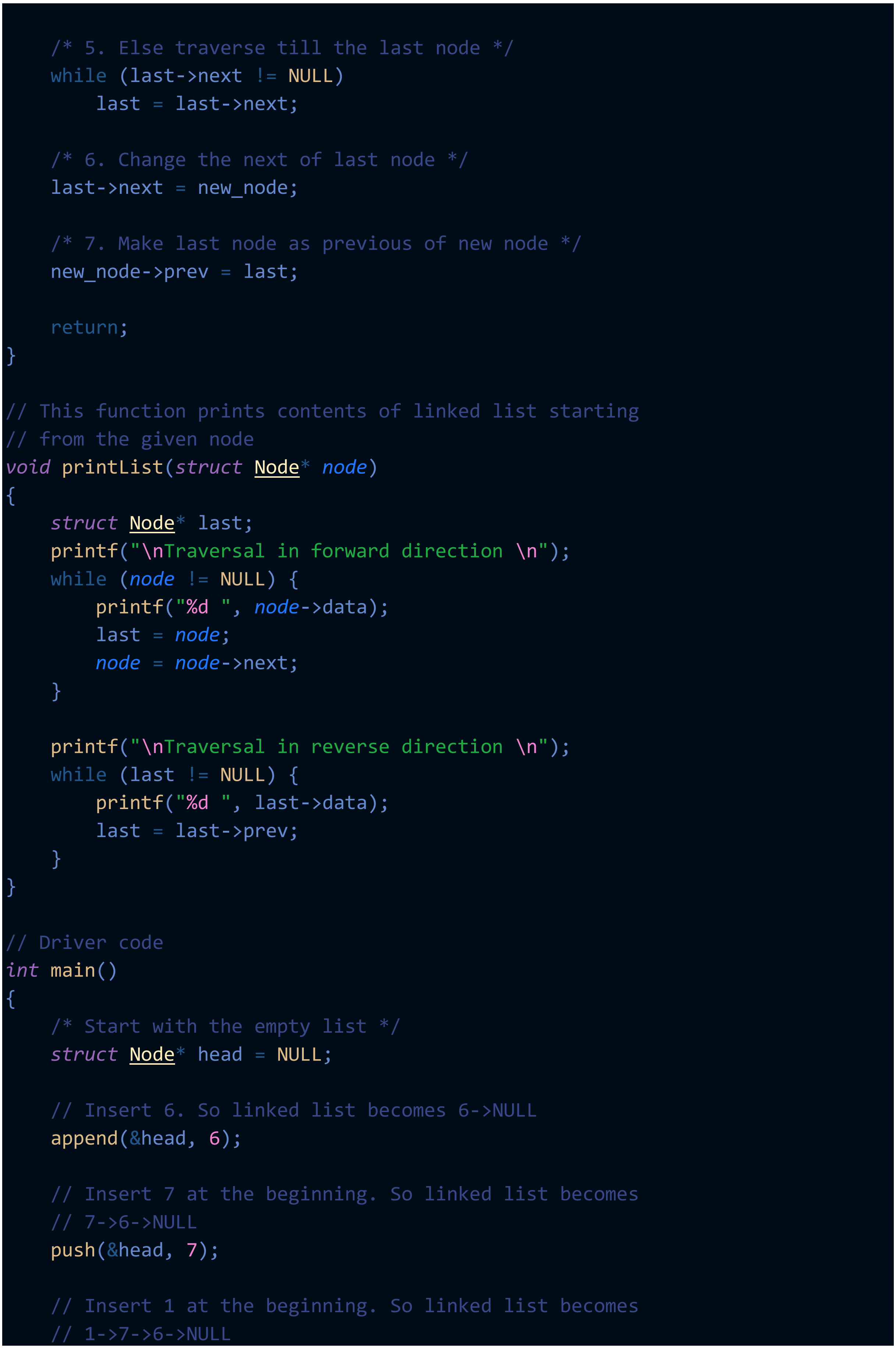
```
*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?1

Enter value?
20

*************************Main Menu*****************************

=================================================================

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?3

printing values .....

20
```

Doubly Linked List:

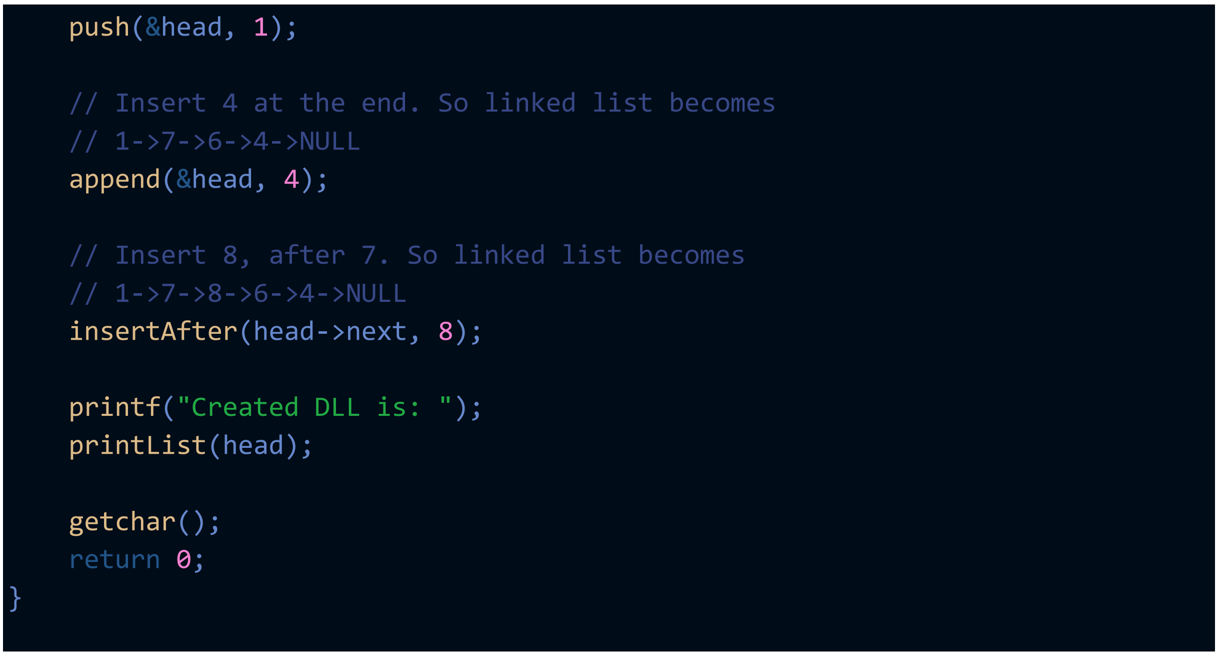
```
// A complete working C program to
// demonstrate all insertion
// methods
#include <stdio.h>
#include <stdlib.h>

// A linked list node
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

/* Given a reference (pointer to pointer) to the head of a
list and an int, inserts a new node on the front of the
list. */
void push(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node
        = (struct Node*)malloc(sizeof(struct Node));

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. Make next of new node as head and previous as NULL
    */
    new_node->next = (*head_ref);
    new_node->prev = NULL;

    /* 4. change prev of head node to new node */
    if ((*head_ref) != NULL)
        (*head_ref)->prev = new_node;

    /* 5. move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Given a node as prev_node, insert a new node after the
* given node */
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL) {
        printf("the given previous node cannot be NULL");
```



```c
        return;
    }

    /* 2. allocate new node */
    struct Node* new_node
        = (struct Node*)malloc(sizeof(struct Node));

    /* 3. put in the data */
    new_node->data = new_data;

    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;

    /* 5. Make the next of prev_node as new_node */
    prev_node->next = new_node;

    /* 6. Make prev_node as previous of new_node */
    new_node->prev = prev_node;

    /* 7. Change previous of new_node's next node */
    if (new_node->next != NULL)
        new_node->next->prev = new_node;
}

/* Given a reference (pointer to pointer) to the head
of a DLL and an int, appends a new node at the end */
void append(struct Node** head_ref, int new_data)
{
    /* 1. allocate node */
    struct Node* new_node
        = (struct Node*)malloc(sizeof(struct Node));

    struct Node* last = *head_ref; /* used in step 5*/

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. This new node is going to be the last node, so
        make next of it as NULL*/
    new_node->next = NULL;

    /* 4. If the Linked List is empty, then make the new
        node as head */
    if (*head_ref == NULL) {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }
```

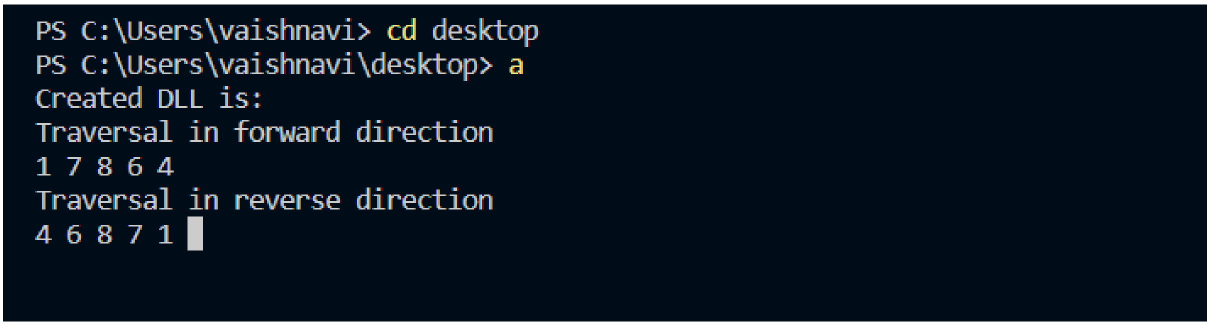


```
    /* 5. Else traverse till the last node */
    while (last->next != NULL)
        last = last->next;

    /* 6. Change the next of last node */
    last->next = new_node;

    /* 7. Make last node as previous of new node */
    new_node->prev = last;

    return;
}

// This function prints contents of linked list starting
// from the given node
void printList(struct Node* node)
{
    struct Node* last;
    printf("\nTraversal in forward direction \n");
    while (node != NULL) {
        printf("%d ", node->data);
        last = node;
        node = node->next;
    }

    printf("\nTraversal in reverse direction \n");
    while (last != NULL) {
        printf("%d ", last->data);
        last = last->prev;
    }
}

// Driver code
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;

    // Insert 6. So linked list becomes 6->NULL
    append(&head, 6);

    // Insert 7 at the beginning. So linked list becomes
    // 7->6->NULL
    push(&head, 7);

    // Insert 1 at the beginning. So linked list becomes
    // 1->7->6->NULL
```



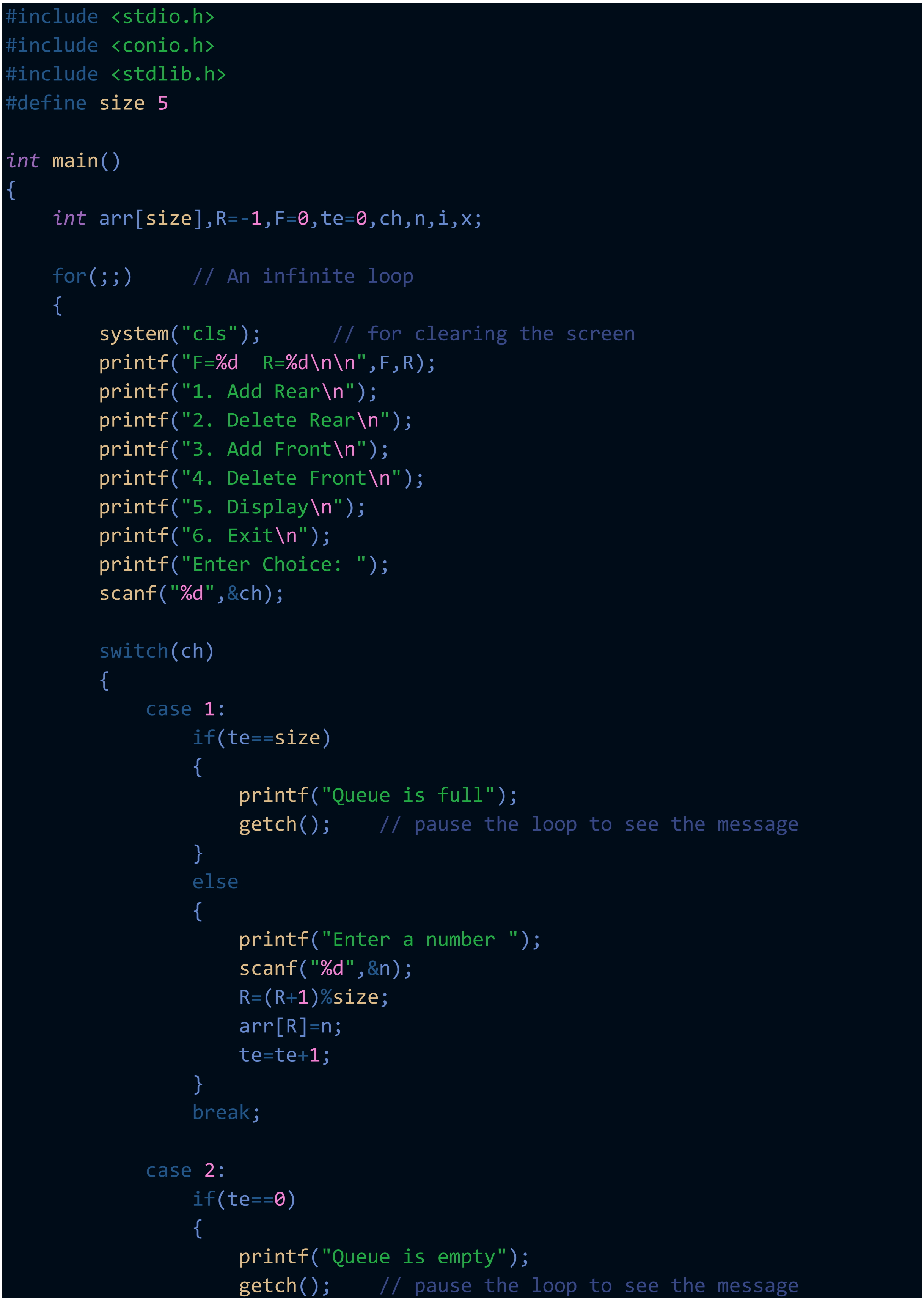
```
    push(&head, 1);

    // Insert 4 at the end. So linked list becomes
    // 1->7->6->4->NULL
    append(&head, 4);

    // Insert 8, after 7. So linked list becomes
    // 1->7->8->6->4->NULL
    insertAfter(head->next, 8);

    printf("Created DLL is: ");
    printList(head);

    getchar();
    return 0;
}
```

Output:

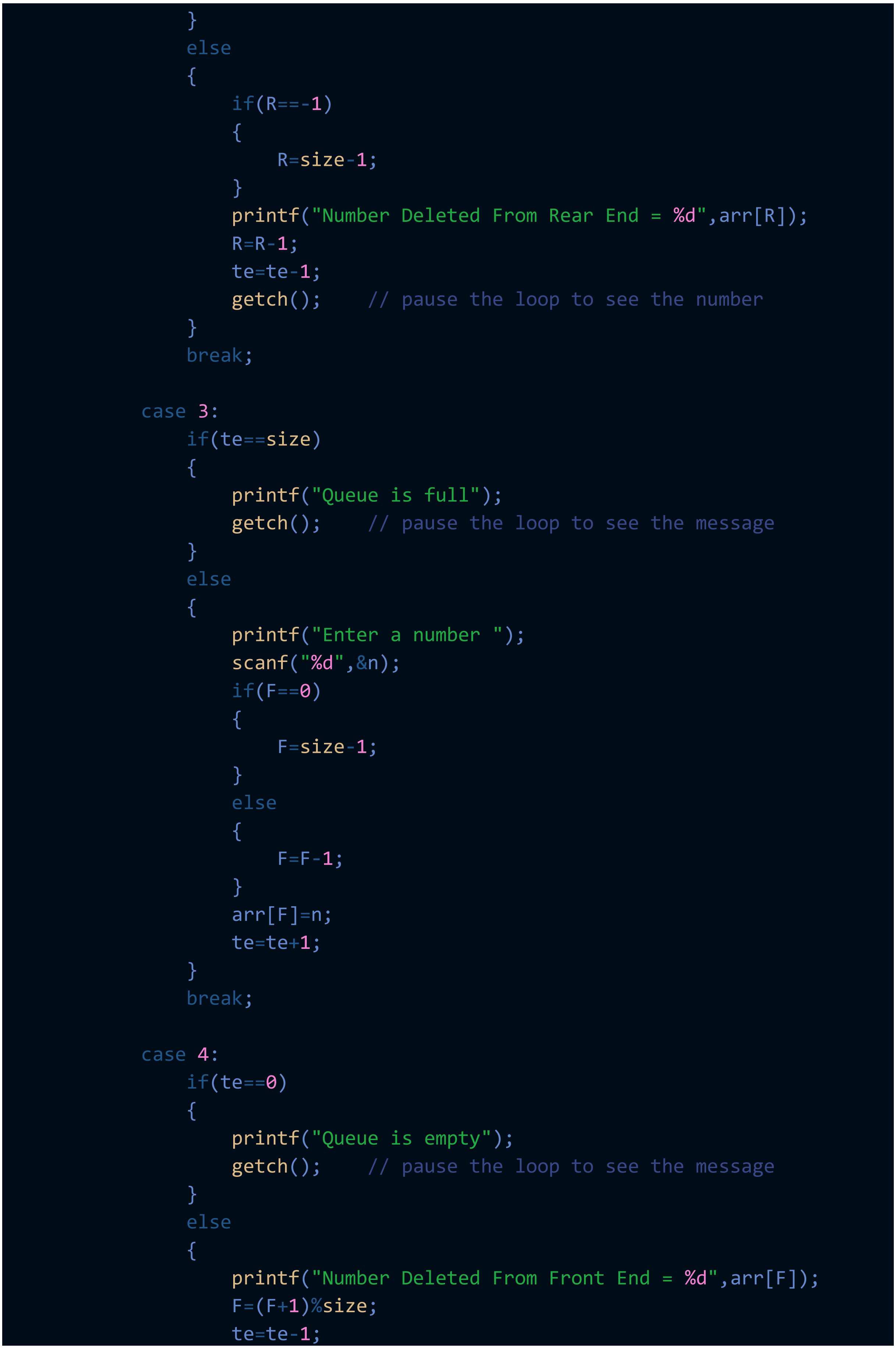


```
PS C:\Users\vaishnavi> cd desktop
PS C:\Users\vaishnavi\desktop> a
Created DLL is:
Traversal in forward direction
1 7 8 6 4
Traversal in reverse direction
4 6 8 7 1
```

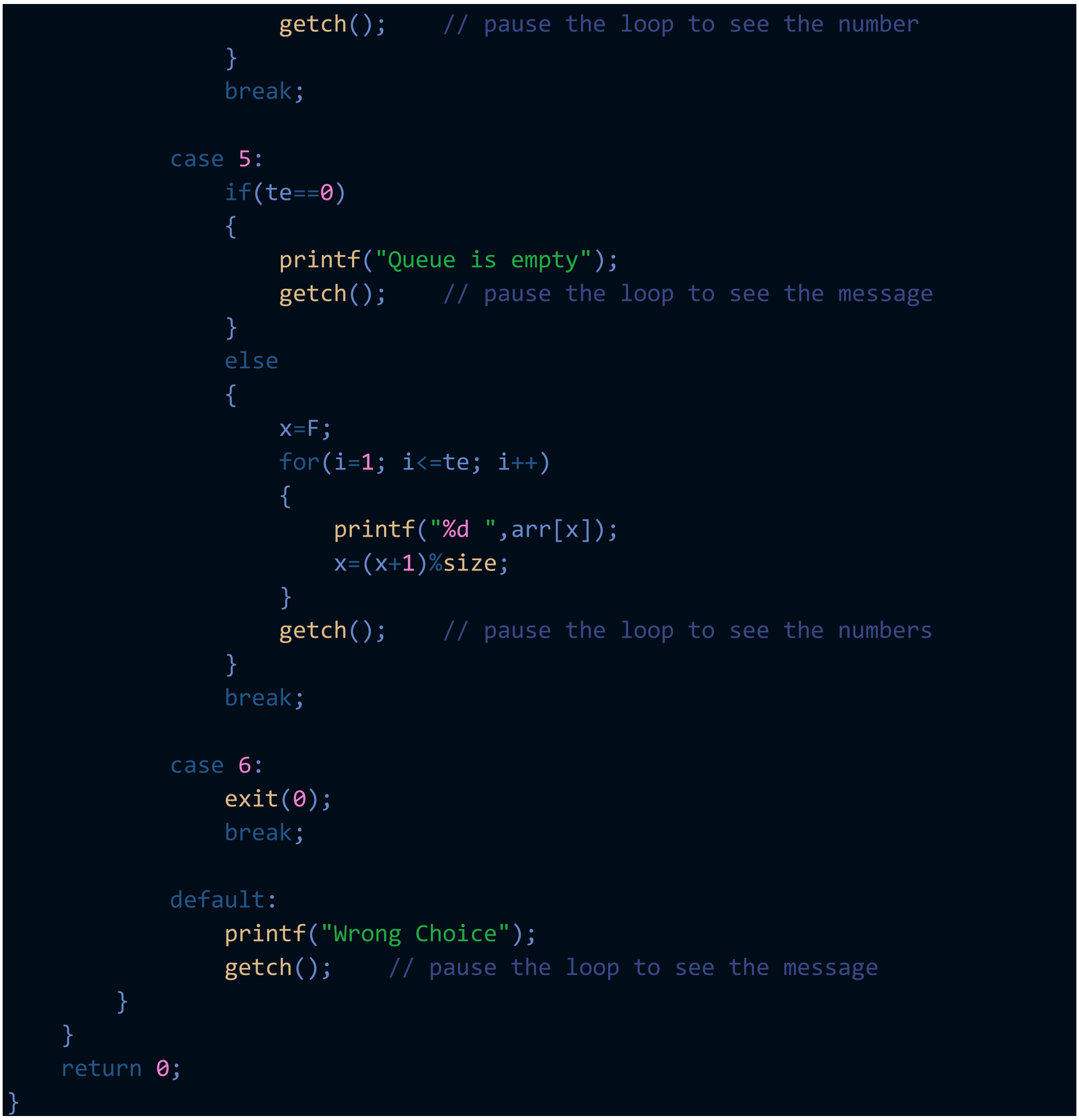
Dequeue:



```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define size 5

int main()
{
    int arr[size],R=-1,F=0,te=0,ch,n,i,x;

    for(;;)     // An infinite loop
    {
        system("cls");      // for clearing the screen
        printf("F=%d  R=%d\n\n",F,R);
        printf("1. Add Rear\n");
        printf("2. Delete Rear\n");
        printf("3. Add Front\n");
        printf("4. Delete Front\n");
        printf("5. Display\n");
        printf("6. Exit\n");
        printf("Enter Choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                if(te==size)
                {
                    printf("Queue is full");
                    getch();    // pause the loop to see the message
                }
                else
                {
                    printf("Enter a number ");
                    scanf("%d",&n);
                    R=(R+1)%size;
                    arr[R]=n;
                    te=te+1;
                }
                break;

            case 2:
                if(te==0)
                {
                    printf("Queue is empty");
                    getch();    // pause the loop to see the message
```

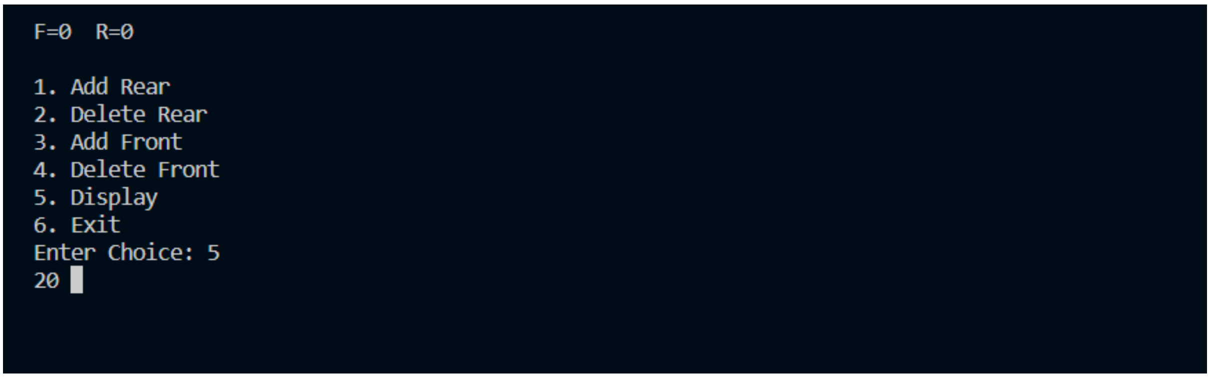


```
                }
                else
                {
                    if(R==-1)
                    {
                        R=size-1;
                    }
                    printf("Number Deleted From Rear End = %d",arr[R]);
                    R=R-1;
                    te=te-1;
                    getch();    // pause the loop to see the number
                }
                break;

            case 3:
                if(te==size)
                {
                    printf("Queue is full");
                    getch();    // pause the loop to see the message
                }
                else
                {
                    printf("Enter a number ");
                    scanf("%d",&n);
                    if(F==0)
                    {
                        F=size-1;
                    }
                    else
                    {
                        F=F-1;
                    }
                    arr[F]=n;
                    te=te+1;
                }
                break;

            case 4:
                if(te==0)
                {
                    printf("Queue is empty");
                    getch();    // pause the loop to see the message
                }
                else
                {
                    printf("Number Deleted From Front End = %d",arr[F]);
                    F=(F+1)%size;
                    te=te-1;
```



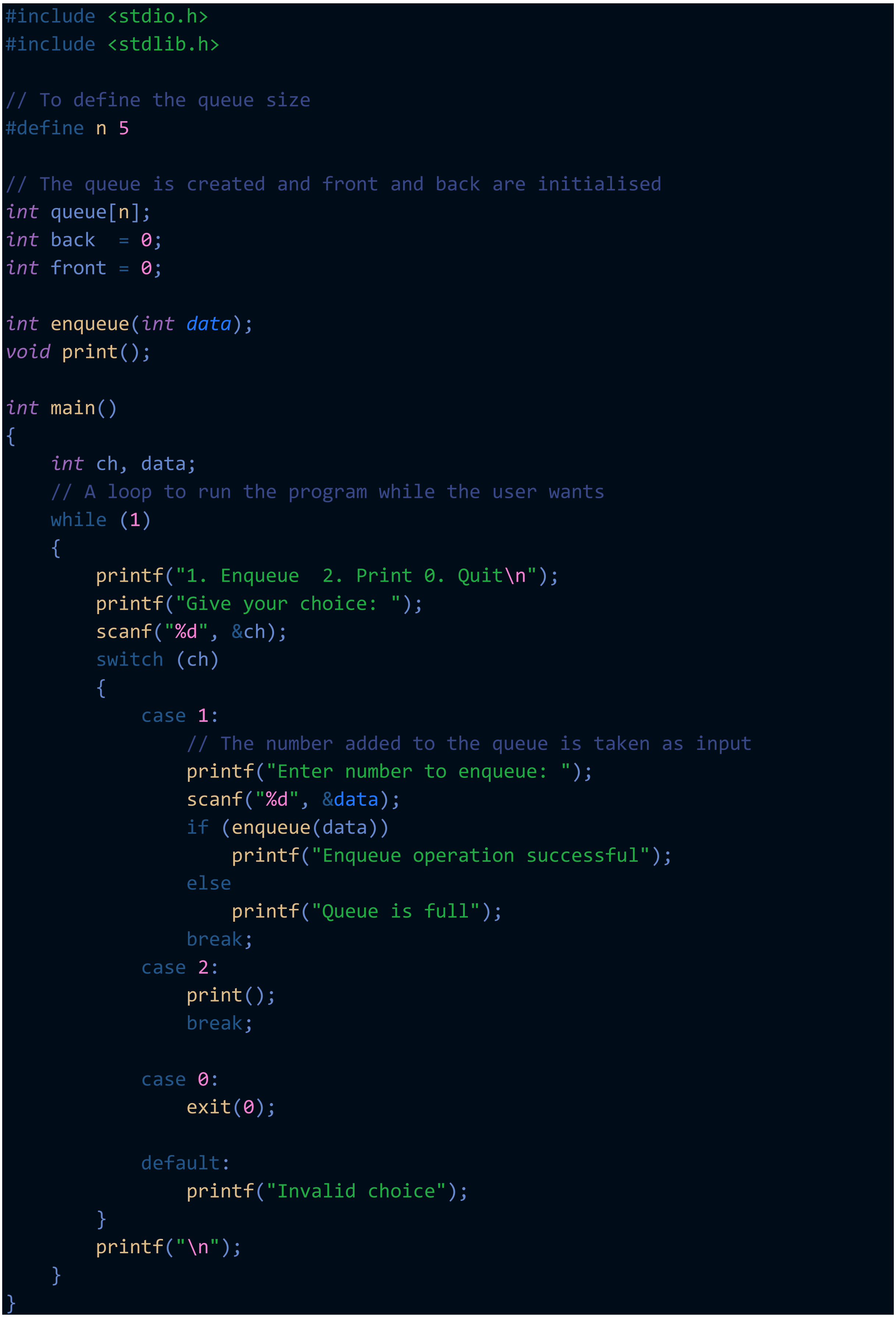
```
                    getch();    // pause the loop to see the number
                }
                break;

            case 5:
                if(te==0)
                {
                    printf("Queue is empty");
                    getch();    // pause the loop to see the message
                }
                else
                {
                    x=F;
                    for(i=1; i<=te; i++)
                    {
                        printf("%d ",arr[x]);
                        x=(x+1)%size;
                    }
                    getch();    // pause the loop to see the numbers
                }
                break;

            case 6:
                exit(0);
                break;

            default:
                printf("Wrong Choice");
                getch();    // pause the loop to see the message
        }
    }
    return 0;
}
```

Output:

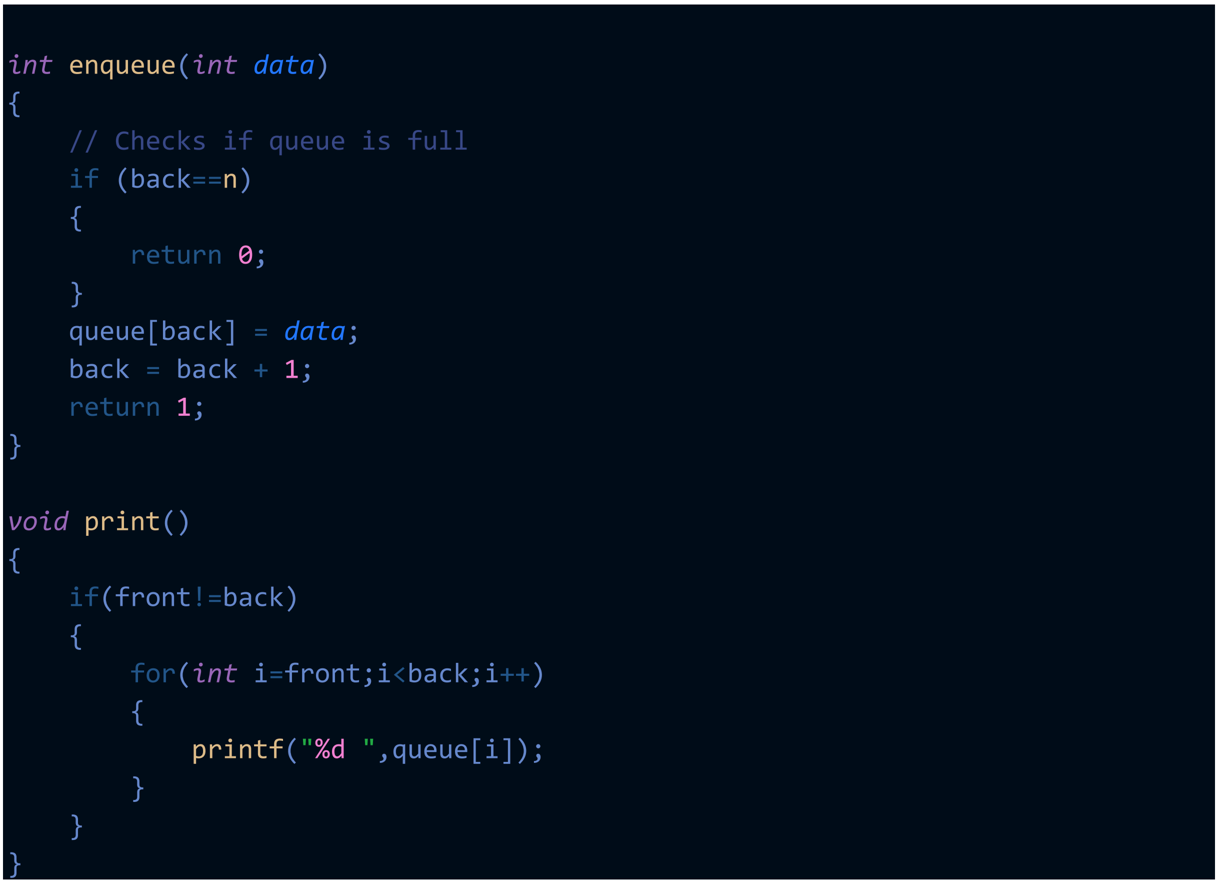


```
F=0  R=0

1. Add Rear
2. Delete Rear
3. Add Front
4. Delete Front
5. Display
6. Exit
Enter Choice: 5
20
```

Enqueue :

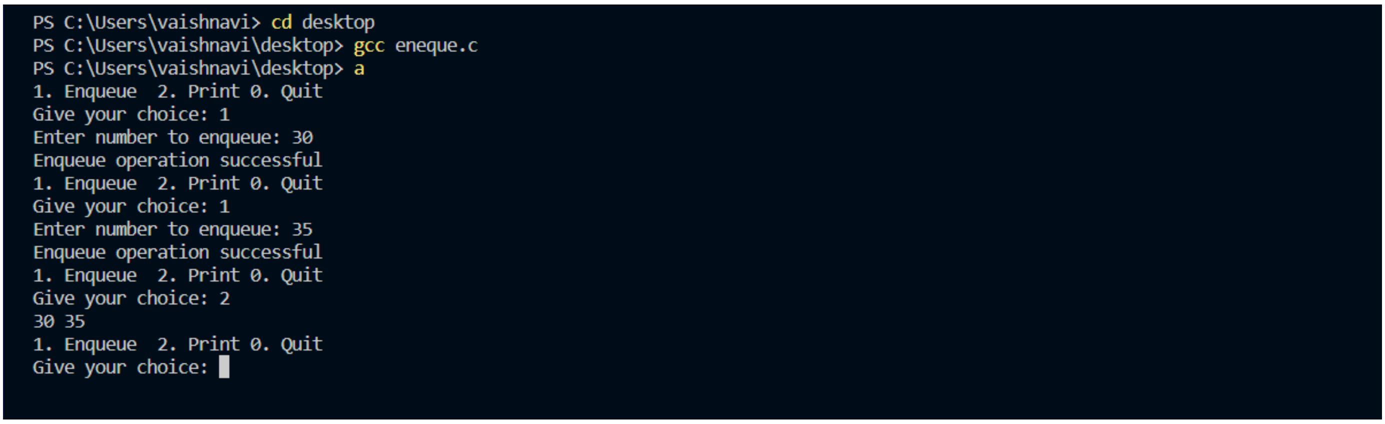


```c
#include <stdio.h>
#include <stdlib.h>

// To define the queue size
#define n 5

// The queue is created and front and back are initialised
int queue[n];
int back   = 0;
int front = 0;

int enqueue(int data);
void print();

int main()
{
    int ch, data;
    // A loop to run the program while the user wants
    while (1)
    {
        printf("1. Enqueue  2. Print 0. Quit\n");
        printf("Give your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
                // The number added to the queue is taken as input
                printf("Enter number to enqueue: ");
                scanf("%d", &data);
                if (enqueue(data))
                    printf("Enqueue operation successful");
                else
                    printf("Queue is full");
                break;
            case 2:
                print();
                break;

            case 0:
                exit(0);

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



```c
int enqueue(int data)
{
    // Checks if queue is full
    if (back==n)
    {
        return 0;
    }
    queue[back] = data;
    back = back + 1;
    return 1;
}

void print()
{
    if(front!=back)
    {
        for(int i=front;i<back;i++)
        {
            printf("%d ",queue[i]);
        }
    }
}
```

Output:



```
PS C:\Users\vaishnavi> cd desktop
PS C:\Users\vaishnavi\desktop> gcc eneque.c
PS C:\Users\vaishnavi\desktop> a
1. Enqueue  2. Print 0. Quit
Give your choice: 1
Enter number to enqueue: 30
Enqueue operation successful
1. Enqueue  2. Print 0. Quit
Give your choice: 1
Enter number to enqueue: 35
Enqueue operation successful
1. Enqueue  2. Print 0. Quit
Give your choice: 2
30 35
1. Enqueue  2. Print 0. Quit
Give your choice: 
```