Data Structure Lab

CEN-391

Program 12

Code :-

```
#include <iostream>
using namespace std;

struct LinkedList
{
    int data;
    LinkedList *next;
    LinkedList *prev;
};

LinkedList *Create_NewNode()
{
    LinkedList *newnode = (LinkedList
*)malloc(sizeof(LinkedList));
    cout << "Enter The Element : ";
    cin >> newnode->data;
    newnode->next = nullptr;
    newnode->prev = nullptr;
    return newnode;
}

void Display(LinkedList *Head, int size)
{
    cout << "Display...\n";
    if (size == 0)
    {
        cout << "Linked List Is Empty!\n";
        return;
    }
    cout << "|Head|";
    while (Head)
    {
        cout << "--|" << Head->data << "|";
        Head = Head->next;
    }
    cout << "--|Tail|\n";
}

void Insert_At_Beginning(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Insert At Beginning Operation Is Selected...
\n";
    LinkedList *newnode = Create_NewNode();
    if (newnode == nullptr)
    {
        cout << "Memory Not Assigned!\n";
        return;
    }
    size++;
    if (Head == nullptr)
    {
        Head = newnode;
        Tail = newnode;
    }
```

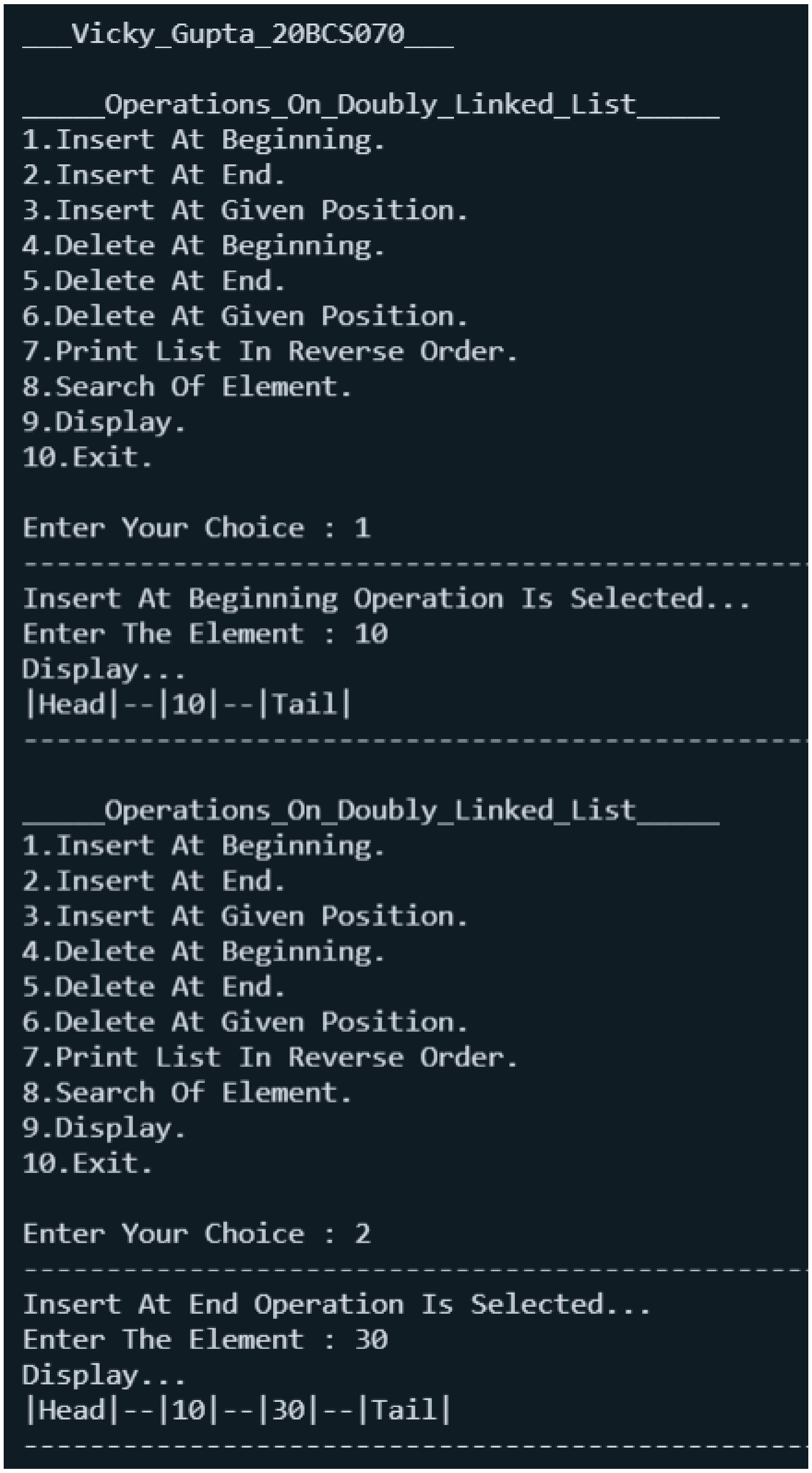
```
        else
        {
            newnode->next = Head;
            Head->prev = newnode;
            Head = newnode;
        }
        Display(Head, size);
    }

    void Insert_At_End(LinkedList *&Head, LinkedList *&Tail, int
    &size)
    {
        cout << "Insert At End Operation Is Selected... \n";
        LinkedList *newnode = Create_NewNode();
        if (size == 0)
        {
            size++;
            Head = newnode;
            Tail = newnode;
            Display(Head, size);
            return;
        }
        if (newnode == nullptr)
        {
            cout << "Memory Not Assigned!\n";
            return;
        }
        size++;
        Tail->next = newnode;
        newnode->prev = Tail;
        Tail = Tail->next;
        Display(Head, size);
    }

    void Insert_At_Given_Position(LinkedList *&Head, LinkedList
    *&Tail, int &size)
    {
        cout << "Insert At Given Position Operation Is
    Selected... \n";
        int k;
        cout << "Enter The Positon Between [0," << size << "] :
    ";
        cin >> k;
        if (k > size || k < 0)
        {
            cout << "Invalid Position!\n";
            return;
        }
        if (k == 0)
            Insert_At_Beginning(Head, Tail, size);
        else if (k == size)
            Insert_At_End(Head, Tail, size);
        else
        {
            size++;
            LinkedList *Current = Head, *newnode =
    Create_NewNode();
            while (k > 1)
            {
                Current = Current->next;
                k--;
            }
            newnode->next = Current->next;
            Current->next->prev = newnode;
            Current->next = newnode;
            newnode->prev = Current;
            Display(Head, size);
        }
    }

    void Delete_At_Beginning(LinkedList *&Head, LinkedList
    *&Tail, int &size)
    {
        cout << "Delete At Beginning Operation Is Selected...
    \n";
        if (size == 0)
        {
            cout << "Linked List Underflow!\n";
```

```
            return;
        }
        size--;
        LinkedList *todelete = Head;
        Head = Head->next;
        if (Head != nullptr)
            Head->prev = nullptr;
        delete todelete;
        if (size == 0)
        {
            Head == nullptr;
            Tail == nullptr;
        }
        Display(Head, size);
    }

    void Delete_At_End(LinkedList *&Head, LinkedList *&Tail, int
    &size)
    {
        cout << "Delete At End Operation Is Selected... \n";
        if (size == 0)
        {
            cout << "Linked List Underflow!\n";
            return;
        }
        size--;
        LinkedList *todelete = Tail;
        Tail = Tail->prev;
        Tail->next = nullptr;
        cout << todelete->data << "\n";
        delete todelete;
        if (size == 0)
        {
            Head == nullptr;
            Tail == nullptr;
        }
        Display(Head, size);
    }
```

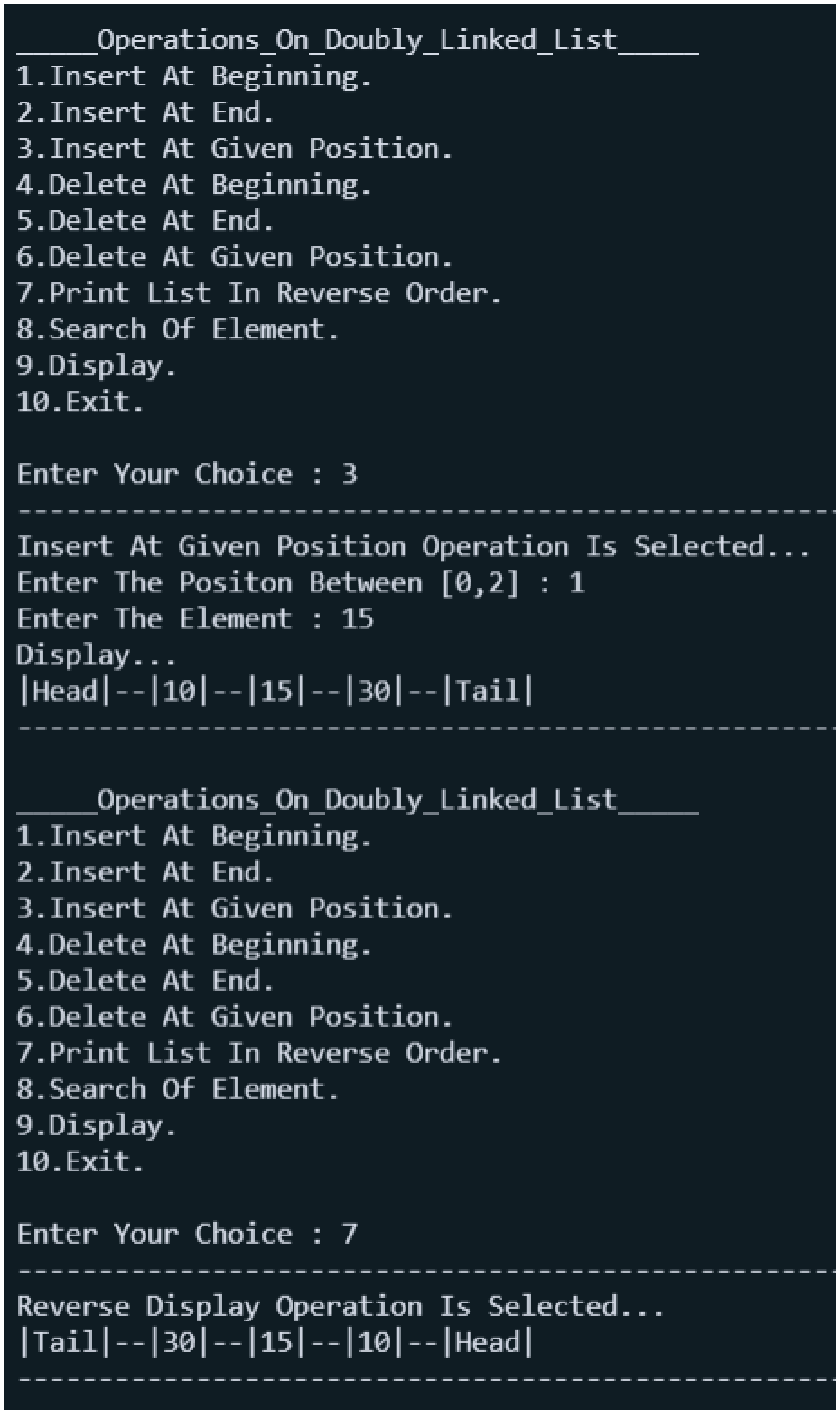
```
void Delete_At_Given_Position(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Delete At Given Position Operation Is
Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    int k;
    cout << "Enter The Positon Between [0," << size - 1 <<
"] : ";
    cin >> k;
    if (k >= size || k < 0)
    {
        cout << "Invalid Position!\n";
        return;
    }
    if (k == 0)
        Delete_At_Beginning(Head, Tail, size);
    else if (k == size - 1)
        Delete_At_End(Head, Tail, size);
    else
    {
        size--;
        LinkedList *Current = Head, *todelete = nullptr;
        while (k > 1)
        {
            Current = Current->next;
            k--;
        }
        todelete = Current->next;
        Current->next = todelete->next;
        todelete->next->prev = Current;
        delete todelete;
        if (size == 0)
        {
            Head == nullptr;
            Tail == nullptr;
        }
        Display(Head, size);
    }
}

void Reverse_Print(LinkedList *Tail, int size)
{
    cout << "Reverse Display Operation Is Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Is Empty!\n";
        return;
    }
    cout << "|Tail|";
    while (Tail)
    {
        cout << "--|" << Tail->data << "|";
        Tail = Tail->prev;
    }
    cout << "--|Head|\n";
}

void Search_Element(LinkedList *Head, int size)
{
    cout << "Search Element Operation Is Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Is Empty!\n";
        return;
    }
    int search;
    cout << "Enter The Element You Want To Search : ";
    cin >> search;
    int isMulti = 0;
    cout << "Do You Want To Search For Single/Multiple
Occurence [0/1] : ";
    cin >> isMulti;
    int Position = 0;
    bool Find = false;
    while (Head)
    {
        if (Head->data == search)
        {
            Find = true;
            cout << Position << " ";
            if (isMulti == false)
                break;
        }
        Position++;
        Head = Head->next;
    }
    if (Find == false)
    {
        cout << "\nElement Not Found!\n";
    }
    else
    {
        cout << "\n"
             << search << " Is Found At Above Positon In
Linked List\n";
    }
}

void Bars()
{
    cout << "-----------------------------------------------
----------------\n";
}

bool Options(LinkedList *&Head, LinkedList *&Tail, int
&size)
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
```

```
    case 1:
        Insert_At_Beginning(Head, Tail, size);
        break;
    case 2:
        Insert_At_End(Head, Tail, size);
        break;
    case 3:
        Insert_At_Given_Position(Head, Tail, size);
        break;
    case 4:
        Delete_At_Beginning(Head, Tail, size);
        break;
    case 5:
        Delete_At_End(Head, Tail, size);
        break;
    case 6:
        Delete_At_Given_Position(Head, Tail, size);
        break;
    case 7:
        Reverse_Print(Tail, size);
        break;
    case 8:
        Search_Element(Head, size);
        break;
    case 9:
        Display(Head, size);
        break;
    case 10:
        return 0;
        break;
    default:
        cout << "Invalid Input!\nTry Again!\n\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "\n_____Operations_On_Doubly_Linked_List_____
\n";
    cout << "1.Insert At Beginning. \n";
    cout << "2.Insert At End. \n";
    cout << "3.Insert At Given Position. \n";
    cout << "4.Delete At Beginning. \n";
    cout << "5.Delete At End. \n";
    cout << "6.Delete At Given Position. \n";
    cout << "7.Print List In Reverse Order. \n";
    cout << "8.Search Of Element. \n";
    cout << "9.Display.\n";
    cout << "10.Exit.\n";
    cout << "\nEnter Your Choice : ";
}

int main()
{
    system("cls");
    cout << "___Vicky_Gupta_20BCS070___\n";
    LinkedList *Head = nullptr, *Tail = nullptr;
    int size = 0;
    while (true)
    {
        Menu();
        if (!Options(Head, Tail, size))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

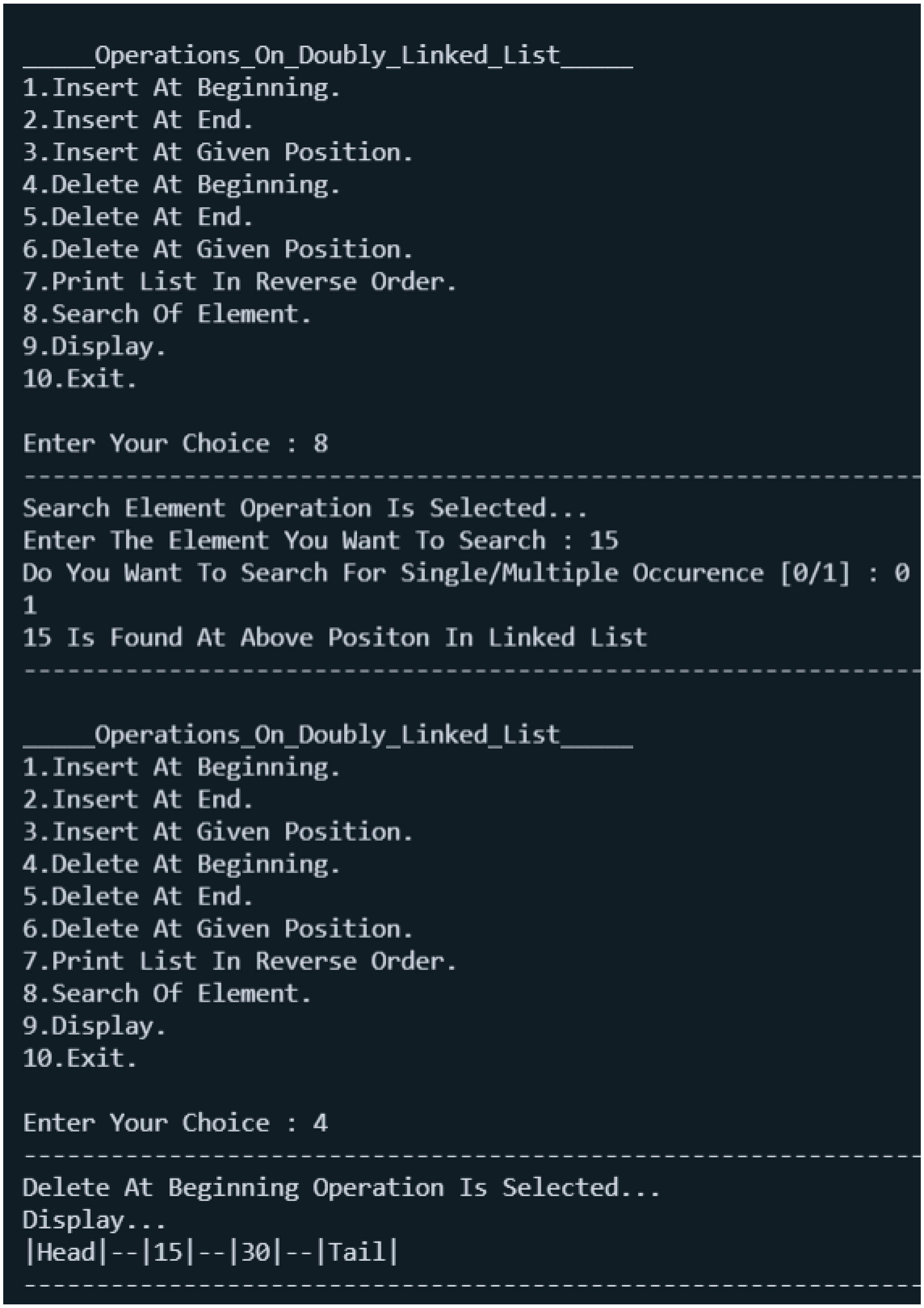
Output :-



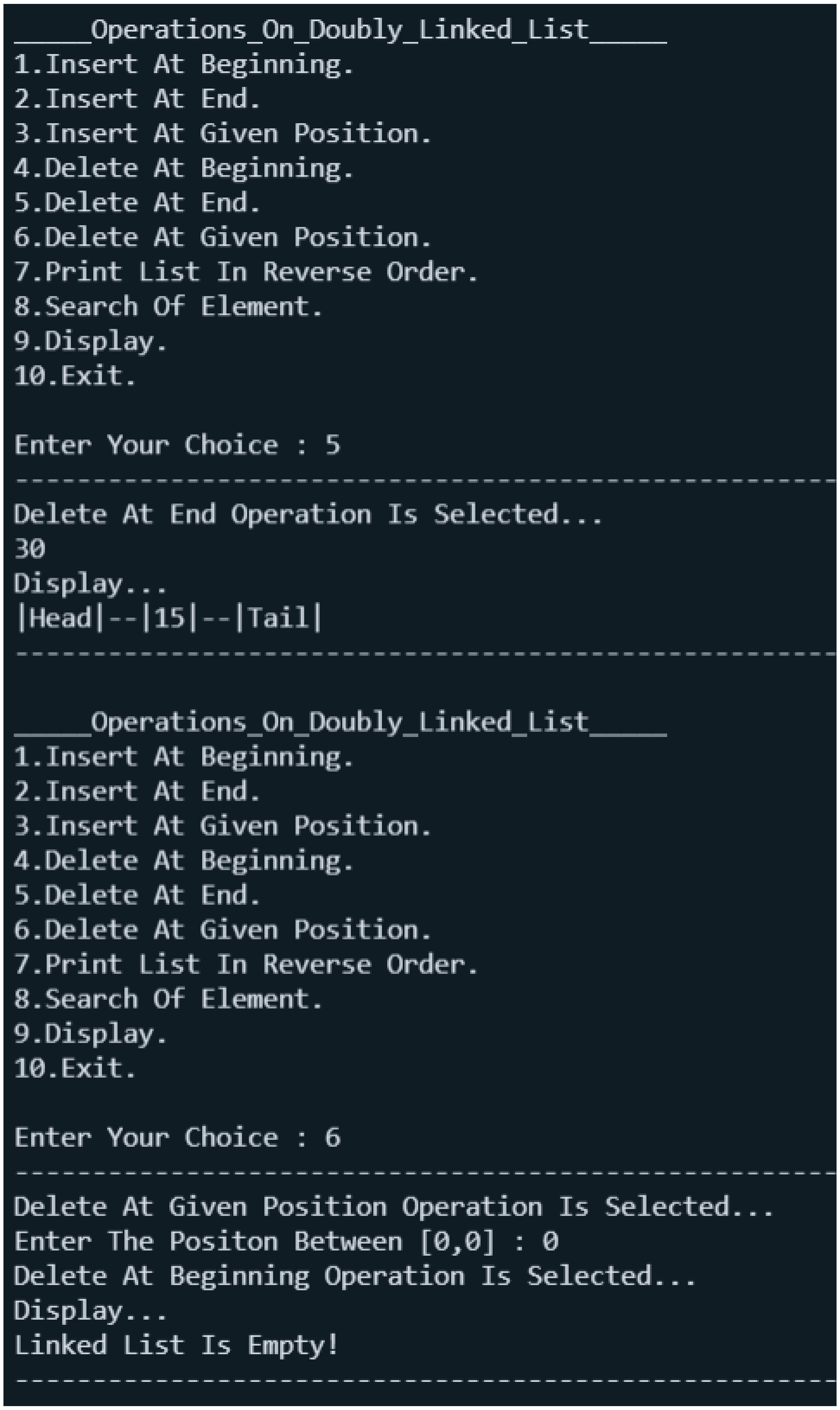
```
___Vicky_Gupta_20BCS070___

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 1
-----------------------------------------------
Insert At Beginning Operation Is Selected...
Enter The Element : 10
Display...
|Head|--|10|--|Tail|
-----------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 2
-----------------------------------------------
Insert At End Operation Is Selected...
Enter The Element : 30
Display...
|Head|--|10|--|30|--|Tail|
-----------------------------------------------
```



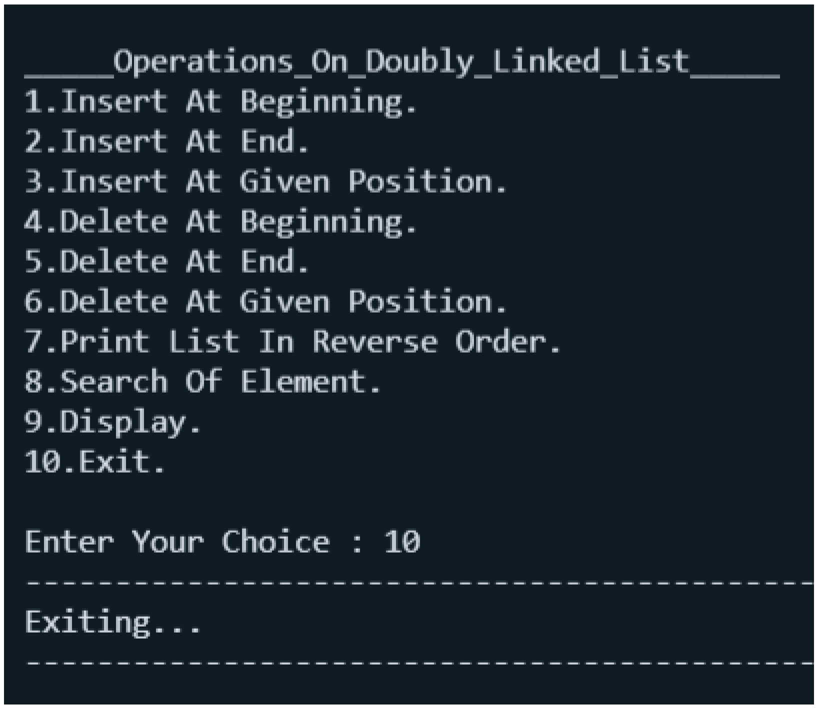
```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 3
--------------------------------------------------
Insert At Given Position Operation Is Selected...
Enter The Positon Between [0,2] : 1
Enter The Element : 15
Display...
|Head|--|10|--|15|--|30|--|Tail|
--------------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 7
--------------------------------------------------
Reverse Display Operation Is Selected...
|Tail|--|30|--|15|--|10|--|Head|
--------------------------------------------------
```



```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 8
--------------------------------------------------------------
Search Element Operation Is Selected...
Enter The Element You Want To Search : 15
Do You Want To Search For Single/Multiple Occurence [0/1] : 0
1
15 Is Found At Above Positon In Linked List
--------------------------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 4
--------------------------------------------------------------
Delete At Beginning Operation Is Selected...
Display...
|Head|--|15|--|30|--|Tail|
--------------------------------------------------------------
```



```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 5
-----------------------------------------------------
Delete At End Operation Is Selected...
30
Display...
|Head|--|15|--|Tail|
-----------------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 6
-----------------------------------------------------
Delete At Given Position Operation Is Selected...
Enter The Positon Between [0,0] : 0
Delete At Beginning Operation Is Selected...
Display...
Linked List Is Empty!
-----------------------------------------------------
```



```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 10
---------------------------------------------
Exiting...
---------------------------------------------
```