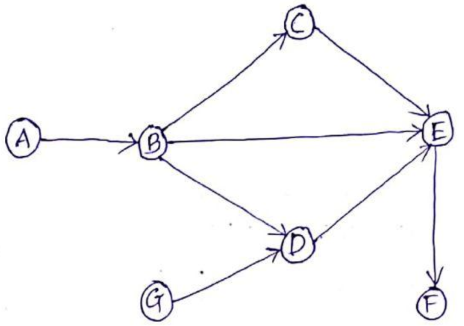
DATA STRUCTURES LAB

External Lab Examination

Brahmaduttan S
TKM20MCA215
Roll no. MCA215
Semester 1
MCA2020-22

Question. 1

Q. consider a directed acyclic graph as given in following figure.



Develop a program to implement topological Sorting.

AIM: To implement Topological Sorting to the given acyclic graph.

ALGORITHM:

Step 1: Start.

Step 2: Read the no of vertexes from the user.

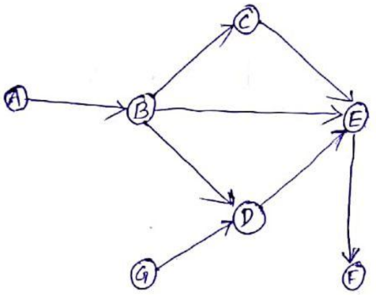
Step 3: Store vertex In degree in an array.

Step 4: Generate a Queue with the Indegree = 0.

Step 5: Remove it from the graph. and repeat

Step 6: Stop.

Procedure



Adjacency Matrix

$$\begin{array}{c|ccccccc}
 & \begin{matrix}1\\(A)\end{matrix} & \begin{matrix}2\\(B)\end{matrix} & \begin{matrix}3\\(C)\end{matrix} & \begin{matrix}4\\(D)\end{matrix} & \begin{matrix}5\\(E)\end{matrix} & \begin{matrix}6\\(F)\end{matrix} & \begin{matrix}7\\(G)\end{matrix} \\
\hline
1\,(A) & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2\,(B) & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
3\,(C) & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
4\,(D) & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
5\,(E) & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
6\,(F) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7\,(G) & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}$$

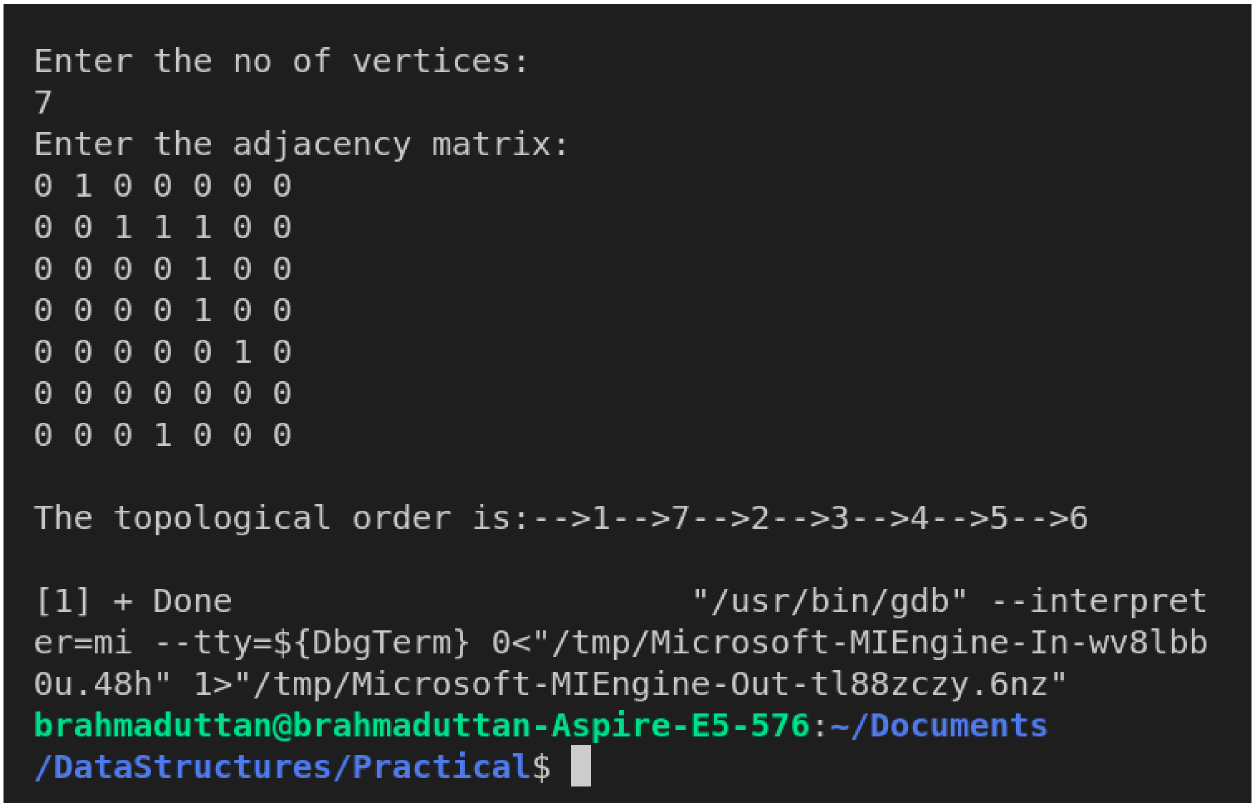
Nodes	In-degree
1 (A)	0
2 (B)	1
3 (C)	1
4 (D)	2
5 (E)	3
6 (F)	1
7 (G)	0

Program Code:

```c
#include <stdio.h>
int main(){
    int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;
    printf("Enter the no of vertices:\n");
    scanf("%d",&n);

    printf("Enter the adjacency matrix:\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    }
    for(i=0;i<n;i++){
        indeg[i]=0;
        flag[i]=0;
    }
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            indeg[i]=indeg[i]+a[j][i];
    printf("\nThe topological order is:");
    while(count<n){
        for(k=0;k<n;k++){
            if((indeg[k]==0) && (flag[k]==0)){
                printf("-->%d",(k+1));
                flag [k]=1;
            }
            for(i=0;i<n;i++){
                if(a[i][k]==1)
                    indeg[k]--;
            }
        }
        count++;
    }
    printf("\n\n");
    return 0;
}
```

Result: Program is successfully executed and output obtained

Output:



```
Enter the no of vertices:
7
Enter the adjacency matrix:
0 1 0 0 0 0 0
0 0 1 1 1 0 0
0 0 0 0 1 0 0
0 0 0 0 1 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 1 0 0 0

The topological order is:-->1-->7-->2-->3-->4-->5-->6

[1] + Done                       "/usr/bin/gdb" --interpret
er=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-wv8lbb
0u.48h" 1>"/tmp/Microsoft-MIEngine-Out-tl88zczy.6nz"
brahmaduttan@brahmaduttan-Aspire-E5-576:~/Documents
/DataStructures/Practical$ 
```

Question 2.

Qn. Wraite a progeam for creating Doubly Linked List and perfoam the following operations.

A) Insert an element at a particular position.

B) Search an element.

c) Delete an element at the end of the list.

AIM: To create a Doubly Linked List and Perform some operations in C

ALGORITHM

```
A) Insert an element at a particular position.
   Step 1: Start
   step 2: Create a node (new node).
           new node -> data = Value.
           new node -> prev = Null
           new node -> next = Null.
  Step 3: Ptr = Start.
            while (ptr -> Next != Choice)
            {
              Ptr = ptr -> next
            }
 step 4:  new node -> next = e{ptr -> next}
 step 5: new node -> Prev = ptr.
 step 6:  e -> PRRV = new node
```

```
Step 7 : ptr -> next = new node
step 8 : exit.
```

B) Search an element

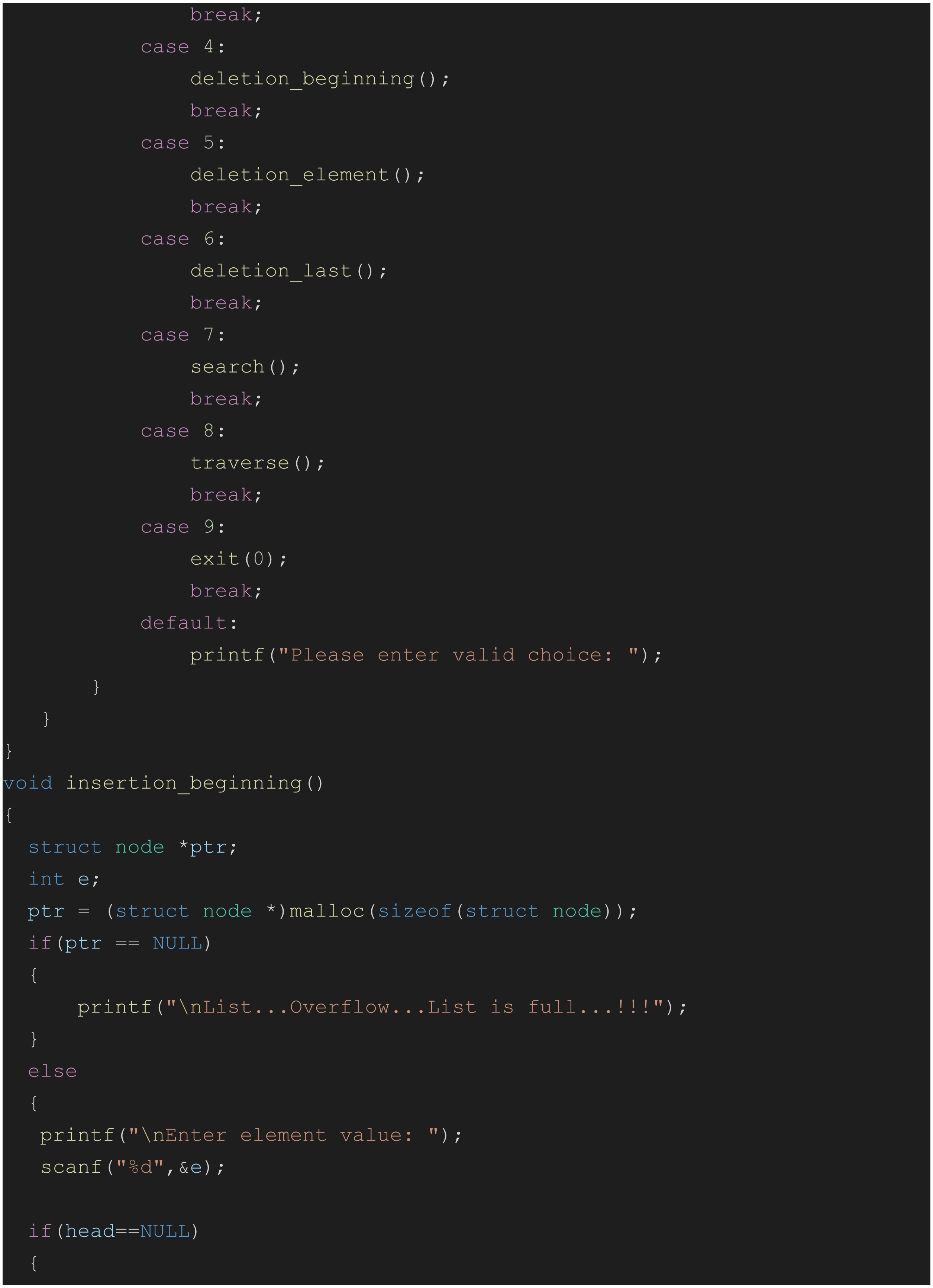
```
Step 1:  Start
Step 2:  ptr = head.
             if (ptr == null)
                  List is empty.
             else
Step 3:  Read elemente from user to do search operation
             while (ptr != null)
             {  if (ptr -> data == e)
                 {
                    Print "Element at location at e.
                 }
                 else
Step 4:          ptr = ptr -> next.
             if (flag == 1)
                 Print "Element not found".
Step 5:  Stop
```

c) Delete the element at the end of the list.

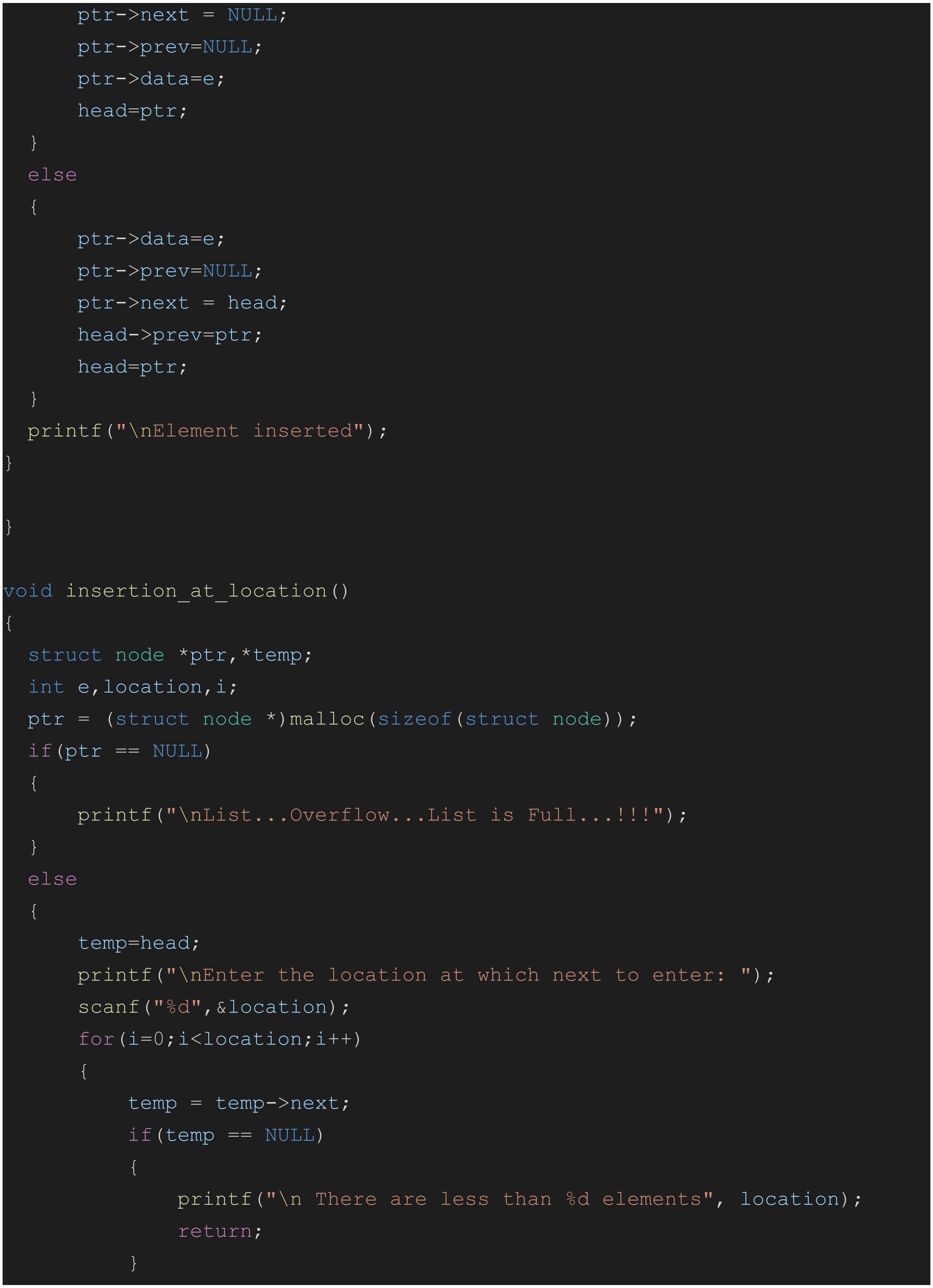
```
Step 1 : Start
Step 2 : ptr = head
Step 3 : while (ptr -> next ! = Null)
           {
              ptr = ptr -> next;
           }
Step 4 : Temp = ptr -> prev.
Step 5 : Temp -> next = Null.
Step 6 : free(ptr)
Step 7 :: exit
```

Program Code:

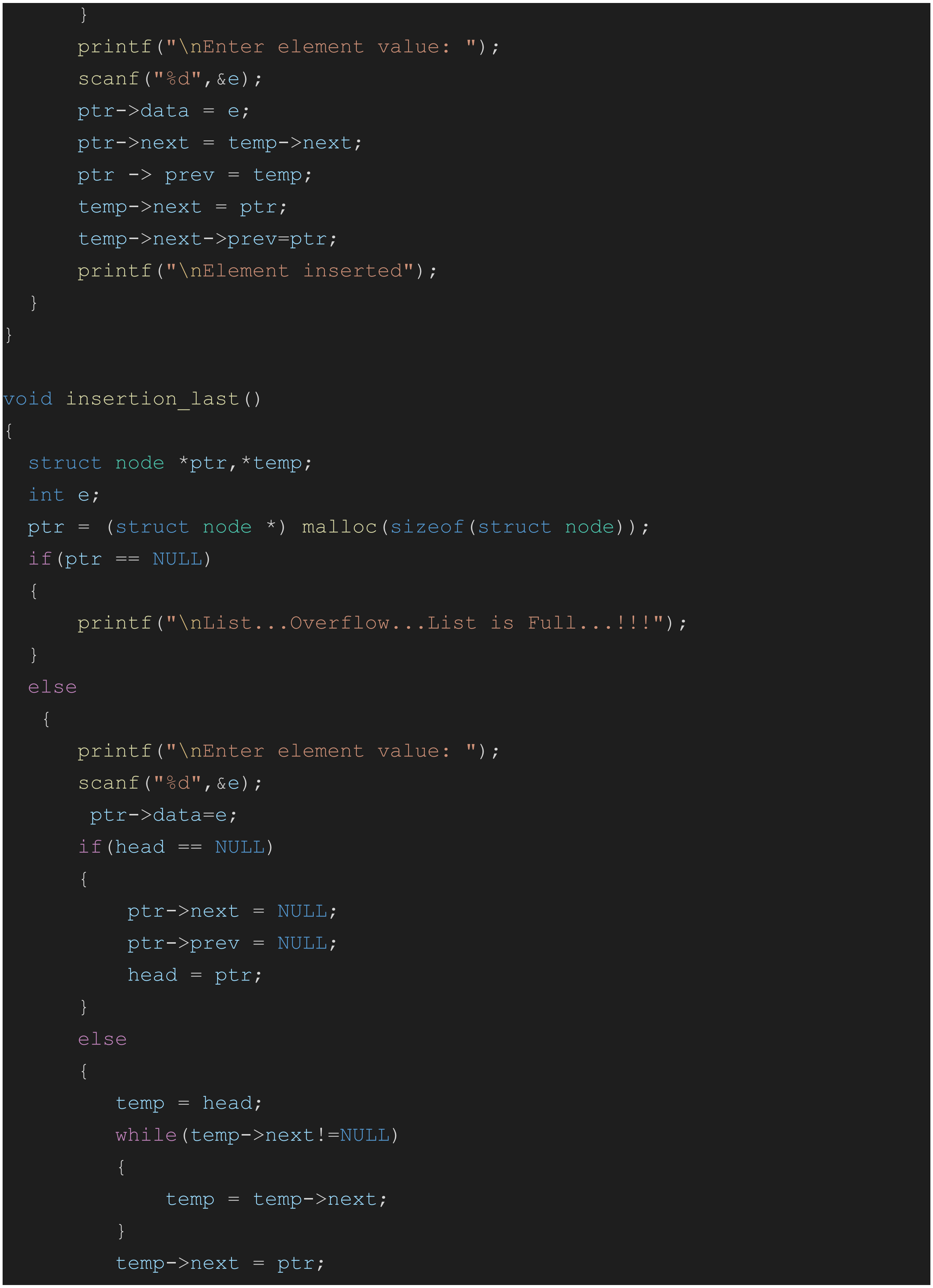
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   struct node *prev;
   struct node *next;
   int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();
void deletion_specified();
void display();
void search();
void main ()
{
int choice =0;
   while(choice != 9)
   {


       printf("\n\n1.Insertion at Beginning\n2.Insertion at a location\n3.Insertion at last\n4.Deletion at Beginning\n5.Delete a specific Element\n6.Deletion at last\n7.Search\n8.Traverse\n9.Exit\n");
       printf("\nEnter your choice: ");
       scanf("%d",&choice);
       switch(choice)
       {
           case 1:
               insertion_beginning();
               break;
           case 2:
               insertion_at_location();
               break;
           case 3:
               insertion_last();
```



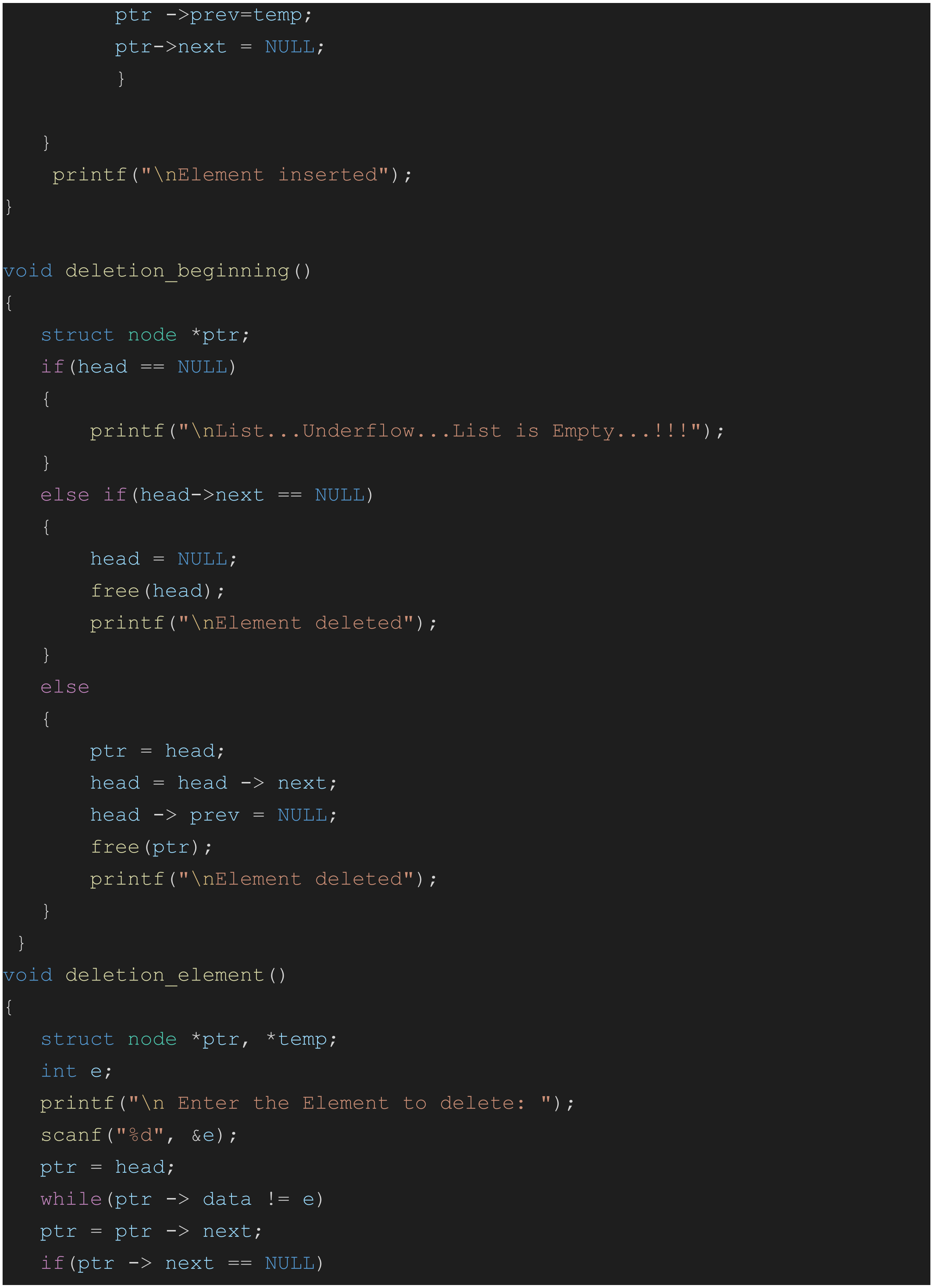
```
               break;
           case 4:
               deletion_beginning();
               break;
           case 5:
               deletion_element();
               break;
           case 6:
               deletion_last();
               break;
           case 7:
               search();
               break;
           case 8:
               traverse();
               break;
           case 9:
               exit(0);
               break;
           default:
               printf("Please enter valid choice: ");
       }
   }
}
void insertion_beginning()
{
  struct node *ptr;
  int e;
  ptr = (struct node *)malloc(sizeof(struct node));
  if(ptr == NULL)
  {
      printf("\nList...Overflow...List is full...!!!");
  }
  else
  {
   printf("\nEnter element value: ");
   scanf("%d",&e);

  if(head==NULL)
  {
```



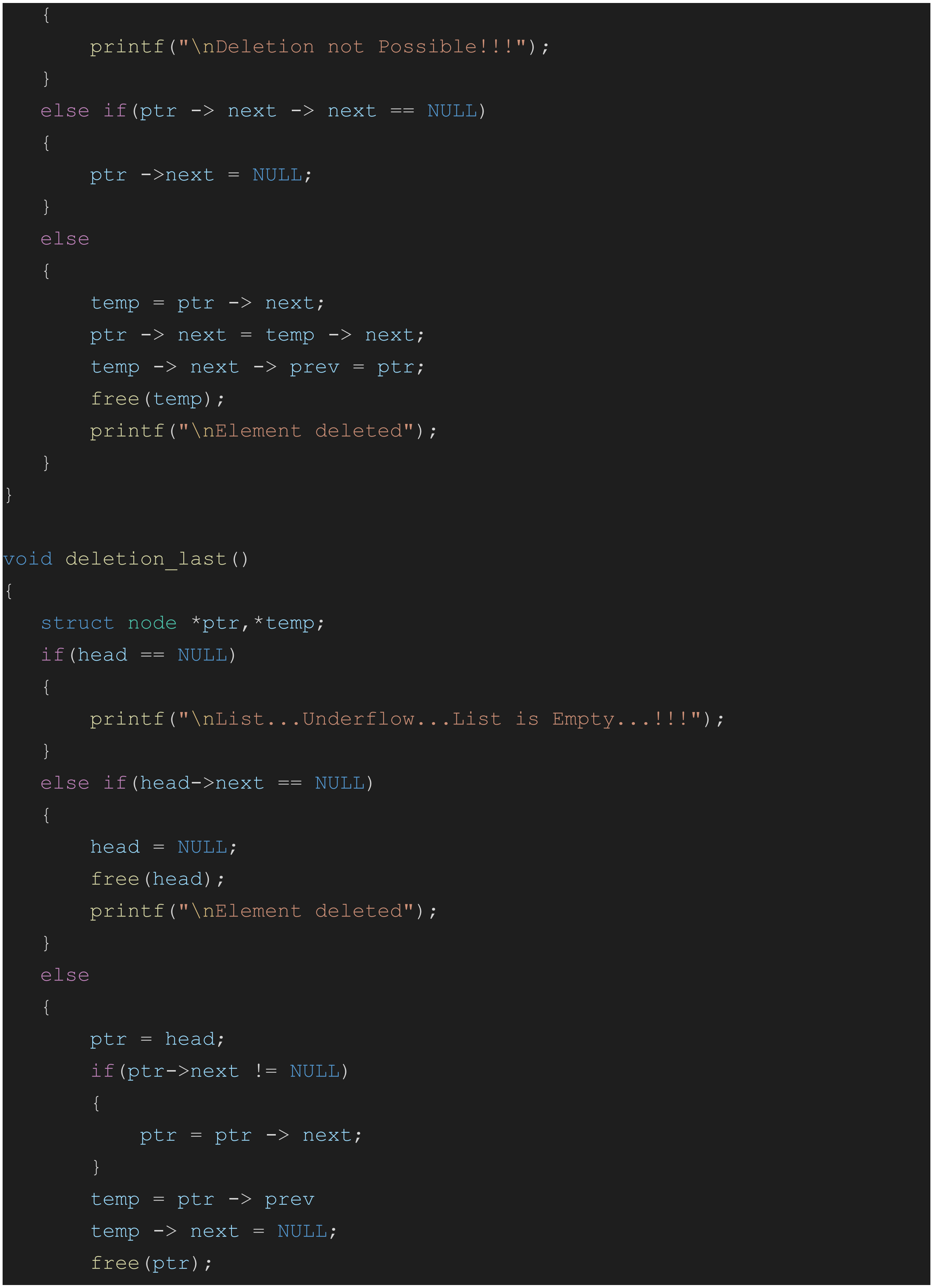
```
      ptr->next = NULL;
      ptr->prev=NULL;
      ptr->data=e;
      head=ptr;
  }
  else
  {
      ptr->data=e;
      ptr->prev=NULL;
      ptr->next = head;
      head->prev=ptr;
      head=ptr;
  }
  printf("\nElement inserted");
}

}

void insertion_at_location()
{
  struct node *ptr,*temp;
  int e,location,i;
  ptr = (struct node *)malloc(sizeof(struct node));
  if(ptr == NULL)
  {
      printf("\nList...Overflow...List is Full...!!!");
  }
  else
  {
      temp=head;
      printf("\nEnter the location at which next to enter: ");
      scanf("%d",&location);
      for(i=0;i<location;i++)
      {
          temp = temp->next;
          if(temp == NULL)
          {
              printf("\n There are less than %d elements", location);
              return;
          }
```



```
      }
      printf("\nEnter element value: ");
      scanf("%d",&e);
      ptr->data = e;
      ptr->next = temp->next;
      ptr -> prev = temp;
      temp->next = ptr;
      temp->next->prev=ptr;
      printf("\nElement inserted");
  }
}

void insertion_last()
{
  struct node *ptr,*temp;
  int e;
  ptr = (struct node *) malloc(sizeof(struct node));
  if(ptr == NULL)
  {
      printf("\nList...Overflow...List is Full...!!!");
  }
  else
   {
      printf("\nEnter element value: ");
      scanf("%d",&e);
       ptr->data=e;
      if(head == NULL)
      {
          ptr->next = NULL;
          ptr->prev = NULL;
          head = ptr;
      }
      else
      {
         temp = head;
         while(temp->next!=NULL)
         {
             temp = temp->next;
         }
         temp->next = ptr;
```



```
         ptr ->prev=temp;
         ptr->next = NULL;
         }

   }
    printf("\nElement inserted");
}

void deletion_beginning()
{
   struct node *ptr;
   if(head == NULL)
   {
       printf("\nList...Underflow...List is Empty...!!!");
   }
   else if(head->next == NULL)
   {
       head = NULL;
       free(head);
       printf("\nElement deleted");
   }
   else
   {
       ptr = head;
       head = head -> next;
       head -> prev = NULL;
       free(ptr);
       printf("\nElement deleted");
   }
 }
void deletion_element()
{
   struct node *ptr, *temp;
   int e;
   printf("\n Enter the Element to delete: ");
   scanf("%d", &e);
   ptr = head;
   while(ptr -> data != e)
   ptr = ptr -> next;
   if(ptr -> next == NULL)
```



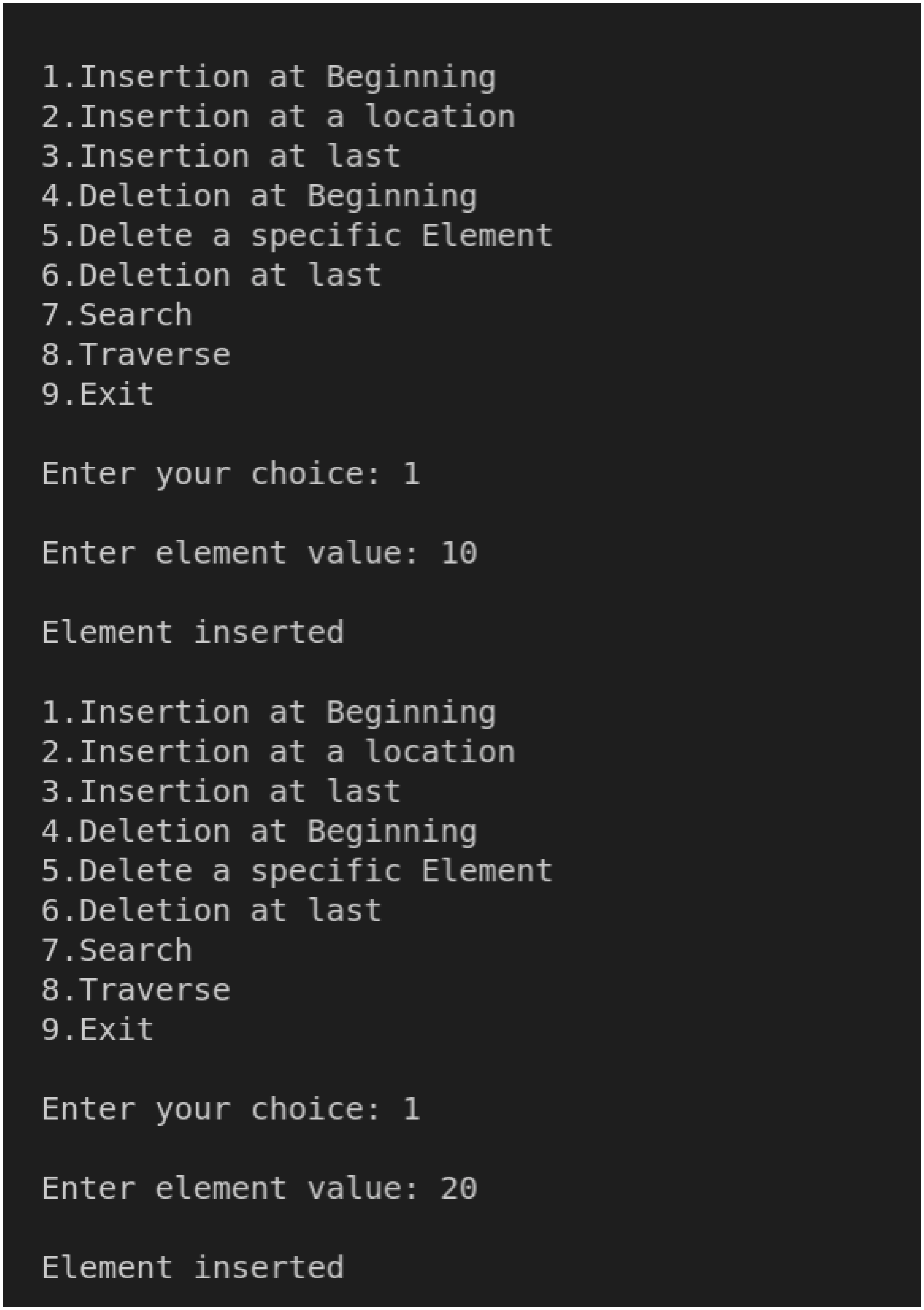
```
    {
        printf("\nDeletion not Possible!!!");
    }
    else if(ptr -> next -> next == NULL)
    {
        ptr ->next = NULL;
    }
    else
    {
        temp = ptr -> next;
        ptr -> next = temp -> next;
        temp -> next -> prev = ptr;
        free(temp);
        printf("\nElement deleted");
    }
}

void deletion_last()
{
    struct node *ptr,*temp;
    if(head == NULL)
    {
        printf("\nList...Underflow...List is Empty...!!!");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nElement deleted");
    }
    else
    {
        ptr = head;
        if(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        temp = ptr -> prev
        temp -> next = NULL;
        free(ptr);
```

```
       printf("\nElement deleted");
   }
}
 void traverse()
{
   struct node *ptr;
   if(head == NULL)
   {
       printf("\nList is Empty...!!!");
   }
   else
   {
       printf("\n Elements in List...");
       ptr = head;
       while(ptr != NULL)
       {
           printf("%d\t",ptr->data);
           ptr=ptr->next;
       }
   }
}

void search()
{
   struct node *ptr;
   int e,i=0,flag;
   ptr = head;
   if(ptr == NULL)
   {
       printf("\nList is Empty...!!!");
   }
   else
   {
       printf("\nEnter the value of element to search: \n");
       scanf("%d",&e);
       while (ptr!=NULL)
       {
           if(ptr->data == e)
           {
               printf("\nElement at location %d ",i);
```

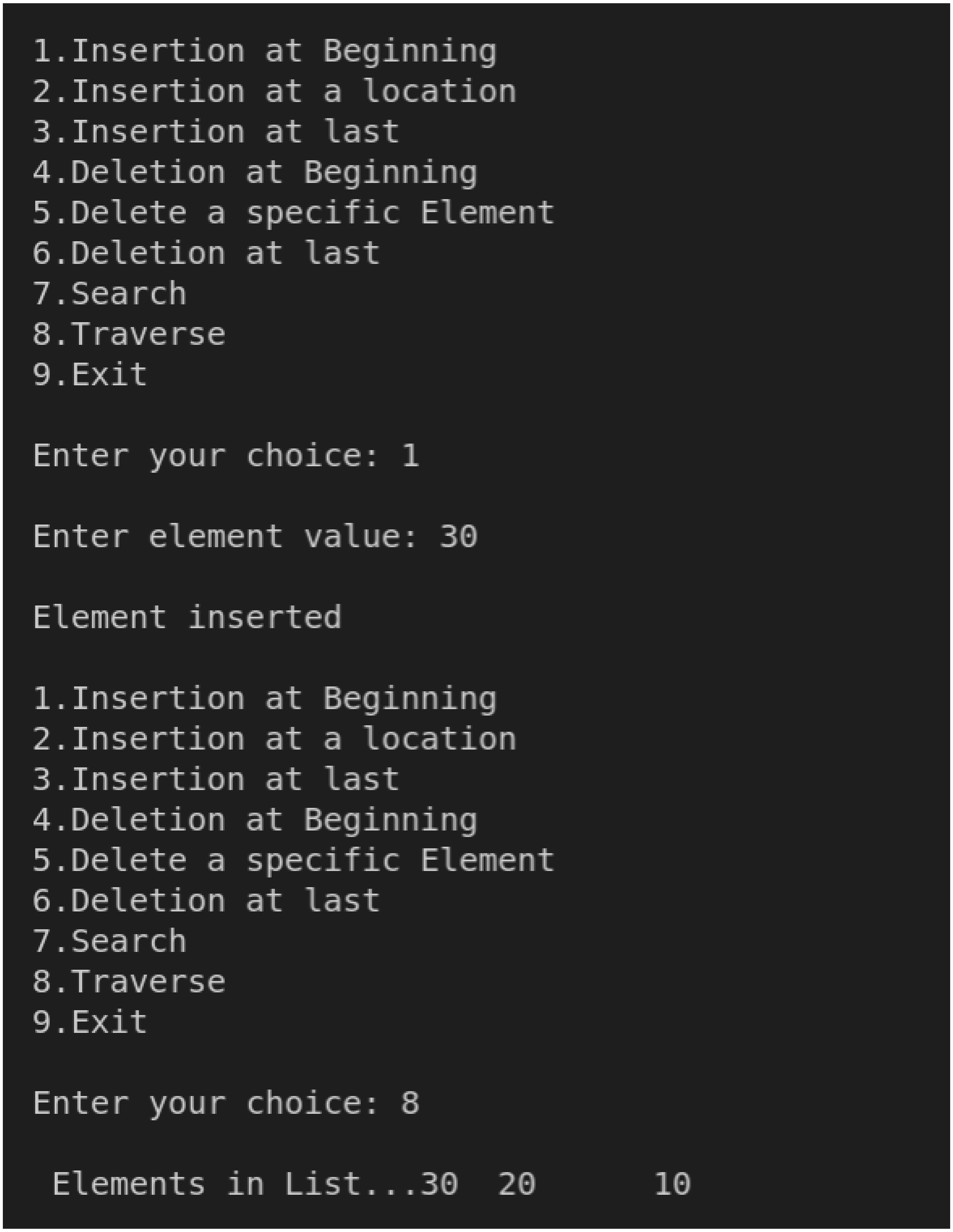
```
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
        if(flag==1)
        {
            printf("\nElement not found\n");
        }
    }

}
```

Result: Program is successfully executed and output obtained

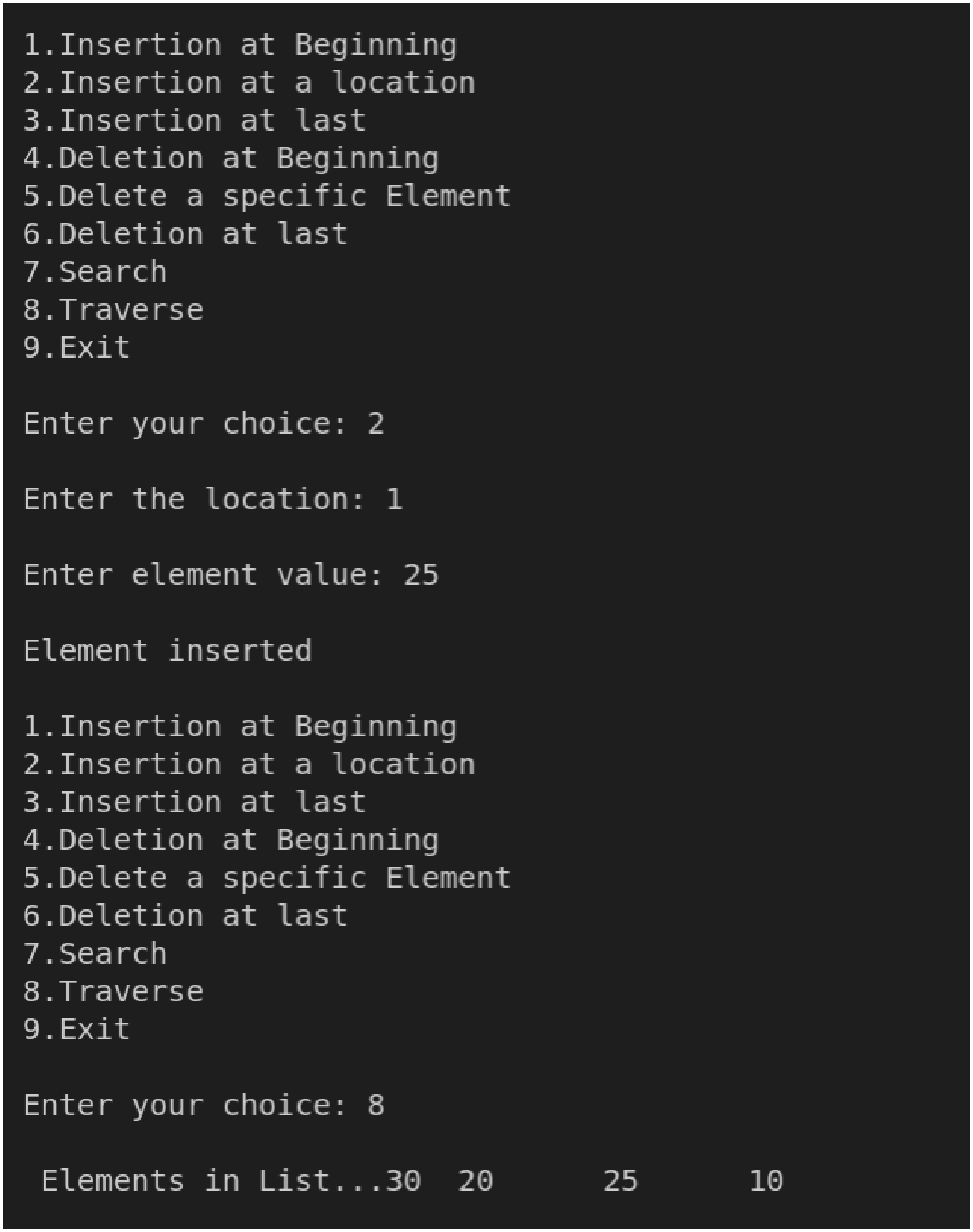
Output:



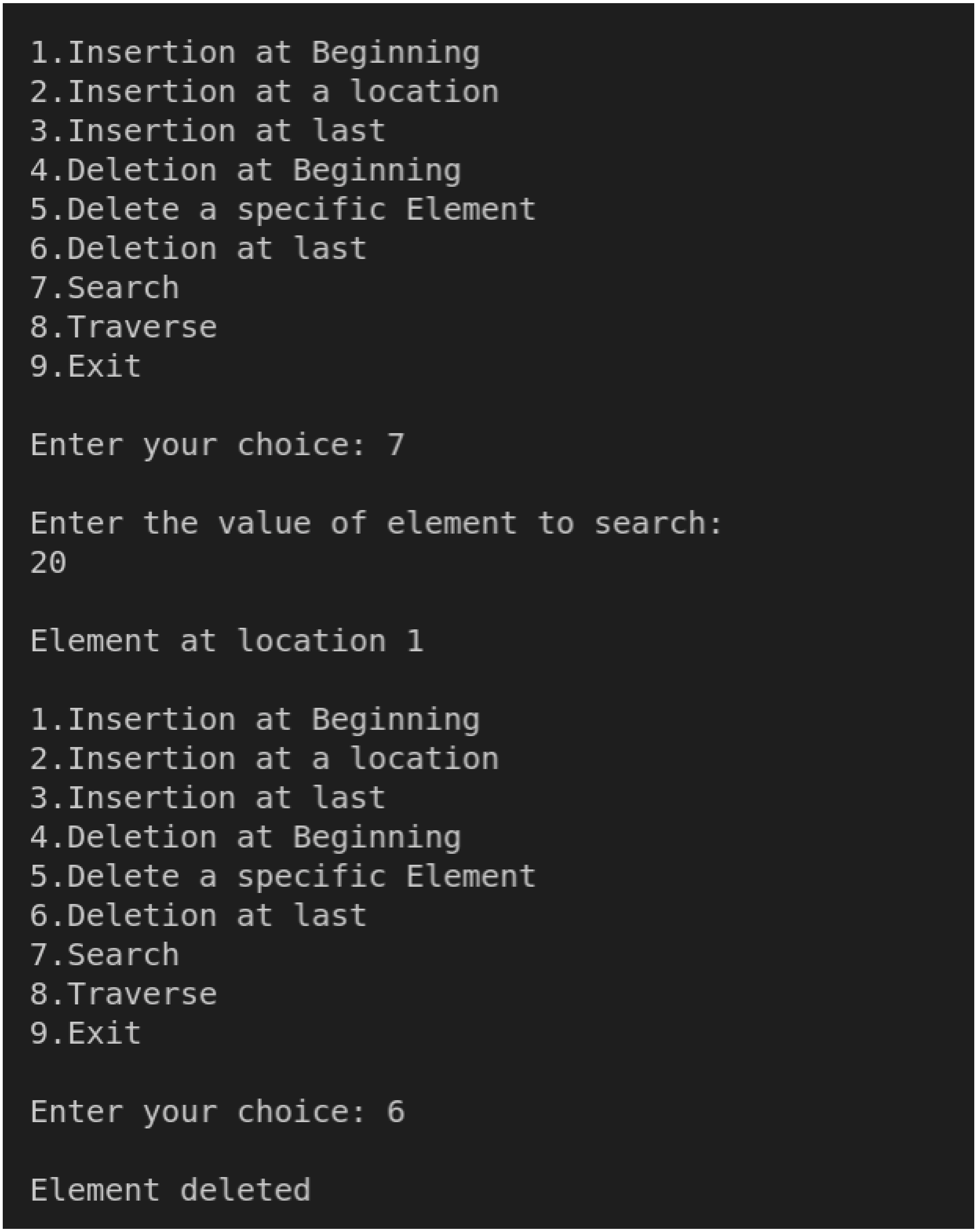
```
1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 1

Enter element value: 10

Element inserted

1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 1

Enter element value: 20

Element inserted
```



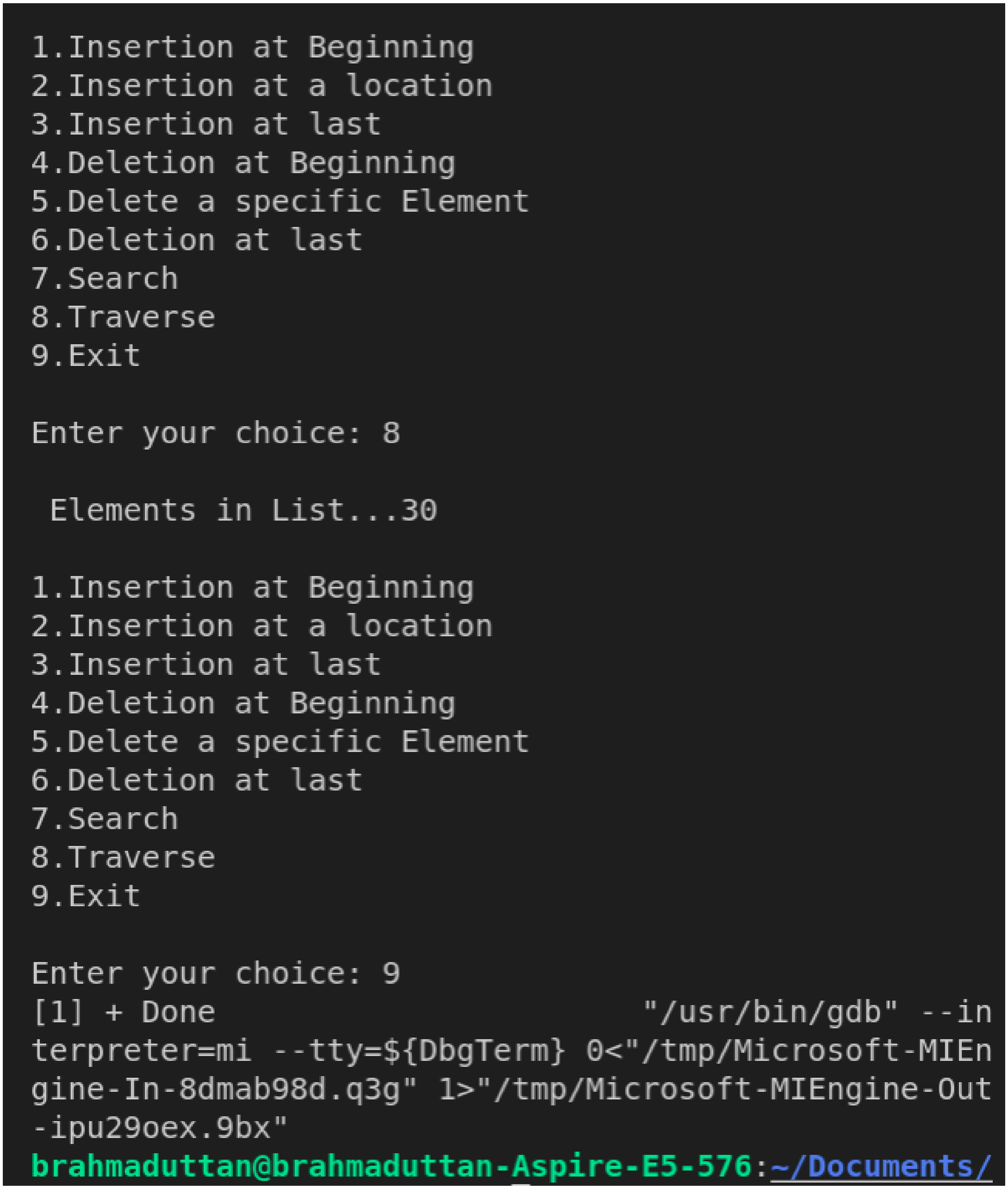
```
1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 1

Enter element value: 30

Element inserted

1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 8

 Elements in List...30  20      10
```



```
1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 2

Enter the location: 1

Enter element value: 25

Element inserted

1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 8

 Elements in List...30  20      25      10
```



```
1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 7

Enter the value of element to search:
20

Element at location 1

1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 6

Element deleted
```



```
1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 8

 Elements in List...30

1.Insertion at Beginning
2.Insertion at a location
3.Insertion at last
4.Deletion at Beginning
5.Delete a specific Element
6.Deletion at last
7.Search
8.Traverse
9.Exit

Enter your choice: 9
[1] + Done                       "/usr/bin/gdb" --in
terpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEn
gine-In-8dmab98d.q3g" 1>"/tmp/Microsoft-MIEngine-Out
-ipu29oex.9bx"
brahmaduttan@brahmaduttan-Aspire-E5-576:~/Documents/
```

GitHub Link :

https://github.com/brahmaduttan/DataStructures/tree/main/EXTERNAL%20LAB%20EXAM%20SEMESTER%201