Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: 3rd - Fall semester / Year:2022
BSc in CSE (Day)

LAB REPORT NO - 03

Course Title: Data Structure Lab
Course Code: CSE 106 / Section: DE-221

Lab Experiment Tittle :

Implement a linked list of Insertion (Beginning, Ending ,At any position) and Deletion

Student Details

	Name	ID
1.	Khondokar Saim	221902353

Lab Date : 01 / 11 / 2022
Submission Date : 17 / 11 / 2022
Course Teacher's Name : Farhana Akter Sunny.

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: ..... Signature: .....
Comments: ..... Date: .....

❑ Tittle of the Lab Experiment :

Implement a linked list of Insertion (Beginning, Ending, At any position) and Deletion

❑ Objectives :

- A linked list is a linear data structure that includes a series of connected nodes. Here, each node stores the data and the address of the next node.
- There are three common types of Linked List.
  - Singly Linked List.
  - Doubly Linked List.
  - Circular Linked List.
- There are various linked list operations that allow us to perform different actions on linked lists. For example, the insertion operation adds a new element to the linked list.
- Here's a list of basic linked list operations that we will cover in this article.
  - Traversal - access each element of the linked list.
  - Insertion - adds a new element to the linked list.
  - Deletion - removes the existing elements.
  - Search - find a node in the linked list.
  - Sort - sort the nodes of the linked list.

1. Implement a linked list of Insertion (Beginning, Ending, At any position) and Deletion

Algorithm :

Step – 1 = Declear function prototype and point all decleared function in the struct structure

Step – 2 = Using switch case in main function (step by step operation)

Step – 3 = Declear particular function on operation Case wise.

Step – 4 = Create All operation function and point them with their same previous defined function in struct

Step – 5 =

```
//Create list function
struct node *create_list(struct node *start)
{
   int i,n,data;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&n);
   start=NULL;
   if(n==0)
      return start;
   printf("Enter the element to be inserted : ");
   scanf("%d",&data);
   start=addtoempty(start,data);

   for(i=2; i<=n; i++)
   {
      printf("Enter the element to be inserted : ");
      scanf("%d",&data);
      start=addatend(start,data);
   }
   return start;
}
```

Step – 6 =

```
//display function
void display(struct node *start)
{
   struct node *p;
   if(start==NULL)
   {
      printf("\nList is empty\n");
      return;
   }
   p=start;
   printf("\nList is :\n");
   while(p!=NULL)
   {
      printf("%d ",p->info);
      p=p->next;
   }
   printf("\n");
}
```

Step – 7 =

```
//insert at begainning function
struct node *addatbeg(struct node *start,int data)
{
   struct node *tmp;
   tmp = (struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   tmp->prev=NULL;
   tmp->next=start;
   start->prev=tmp;
   start=tmp;
   return start;
}
```

Step – 8 =

```
//insert at ending function
struct node *addatend(struct node *start,int data)
{
   struct node *tmp,*p;
   tmp=(struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   p=start;
   while(p->next!=NULL)
      p=p->next;
   p->next=tmp;
   tmp->next=NULL;
   tmp->prev=p;
   return start;
}
```

Step – 9 =

```
//Insertion At Any point
//add after function
struct node *addafter(struct node *start,int data,int item)
{
   struct node *tmp,*p;
   tmp=(struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   p=start;
   while(p!=NULL)
   {
      if(p->info==item)
      {
         tmp->prev=p;
         tmp->next=p->next;
         if(p->next!=NULL)
            p->next->prev=tmp;
         p->next=tmp;
         return start;
      }
      p=p->next;
   }
   printf("\n%d not present in the list\n\n",item);
   return start;
}
```

Step – 10 =

```
//add before function
struct node *addbefore(struct node *start,int data,int item)
{
   struct node *tmp,*q;
   if(start==NULL )
   {
      printf("\nList is empty\n");
      return start;
   }
   if(start->info==item)
   {
      tmp = (struct node *)malloc(sizeof(struct node));
      tmp->info=data;
      tmp->prev=NULL;
      tmp->next=start;
      start->prev=tmp;
      start=tmp;
      return start;
   }
```

```
q=start;
   while(q!=NULL)
   {
      if(q->info==item)
      {
         tmp=(struct node *)malloc(sizeof(struct node));
         tmp->info=data;
         tmp->prev=q->prev;
         tmp->next = q;
         q->prev->next=tmp;
         q->prev=tmp;
         return start;
      }
      q=q->next;
   }
   printf("\n%d not present in the list\n",item);
   return start;
}
```

Step – 11 =

```
//Delete function
struct node *del(struct node *start,int data)
{
   struct node *tmp;
   if(start==NULL)
   {
      printf("\nList is empty\n");
      return start;
   }
   if(start->next==NULL)
      if(start->info==data)
      {
         tmp=start;
         start=NULL;
         free(tmp);
         return start;
      }
      else
      {
         printf("\nElement %d not found\n",data);
         return start;
      }
```

```
/*Deletion of first node*/
   if(start->info==data)
   {
      tmp=start;
      start=start->next;
      start->prev=NULL;
      free(tmp);
      return start;
   }
   /*Deletion in between*/
   tmp=start->next;
   while(tmp->next!=NULL )
   {
      if(tmp->info==data)
      {
         tmp->prev->next=tmp->next;
         tmp->next->prev=tmp->prev;
         free(tmp);
         return start;
      }
      tmp=tmp->next;
   }
   /*Deletion of last node*/
   if(tmp->info==data)
   {
      tmp->prev->next=NULL;
      free(tmp);
      return start;
   }
   printf("\nElement %d not found\n",data);
   return start;
}
```

Code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   struct node *prev;
   int info;
   struct node *next;
};

struct node *create_list(struct node *start);
void display(struct node *start);
struct node *addtoempty(struct node *start,int data);
struct node *addatbeg(struct node *start,int data);
struct node *addatend(struct node *start,int data);
struct node *addafter(struct node *start,int data,int item);
struct node *addbefore(struct node *start,int data,int item);
struct node *del(struct node *start,int data);


int main()
{
   int choice,data,item;
   struct node *start=NULL;
   while(1)
   {
      printf("\n[----LINKED LIST OPERATION----]\n");
      printf("\n1.Create List\n");
      printf("2.Display\n");

      printf("\n----INSERTION PART----\n");

      printf("\n3.Insert at beginning\n");
      printf("4.Insert at end\n");

      printf("\n>>>Insertion at any point<<<\n");
      printf("\n5.Add after\n");
      printf("6.Add before\n");

      printf("\n----DELETION PART----\n");
      printf("\n7.Delete\n");
      //printf("9.Reverse\n");
      printf("8.Quit\n");
```

```
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
    case 1:
       start=create_list(start);
       break;
    case 2:
       display(start);
       break;
    case 3:
       printf("Enter the element to be inserted : ");
       scanf("%d",&data);
       start=addatbeg(start,data);
       break;
    case 4:
       printf("Enter the element to be inserted : ");
       scanf("%d",&data);
       start=addatend(start,data);
       break;
    case 5:
       printf("Enter the element to be inserted : ");
       scanf("%d",&data);
       printf("Enter the element after which to insert : ");
       scanf("%d",&item);
       start=addafter(start,data,item);
       break;
    case 6:
       printf("Enter the element to be inserted : ");
       scanf("%d",&data);
       printf("Enter the element before which to insert : ");
       scanf("%d",&item);
       start=addbefore(start,data,item);
       break;
    case 7:
       printf("Enter the element to be deleted : ");
       scanf("%d",&data);
       start=del(start,data);
       break;
    case 8:
       exit(1);
    default:
       printf("Wrong choice\n");
    }
}
```

```
        return 0;

}

//Create list function
struct node *create_list(struct node *start)
{
   int i,n,data;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&n);
   start=NULL;
   if(n==0)
      return start;
   printf("Enter the element to be inserted : ");
   scanf("%d",&data);
   start=addtoempty(start,data);

   for(i=2; i<=n; i++)
   {
      printf("Enter the element to be inserted : ");
      scanf("%d",&data);
      start=addatend(start,data);
   }
   return start;
}

//display function
void display(struct node *start)
{
   struct node *p;
   if(start==NULL)
   {
      printf("\nList is empty\n");
      return;
   }
   p=start;
   printf("\nList is :\n");
   while(p!=NULL)
   {
      printf("%d ",p->info);
      p=p->next;
   }
   printf("\n");
}
```

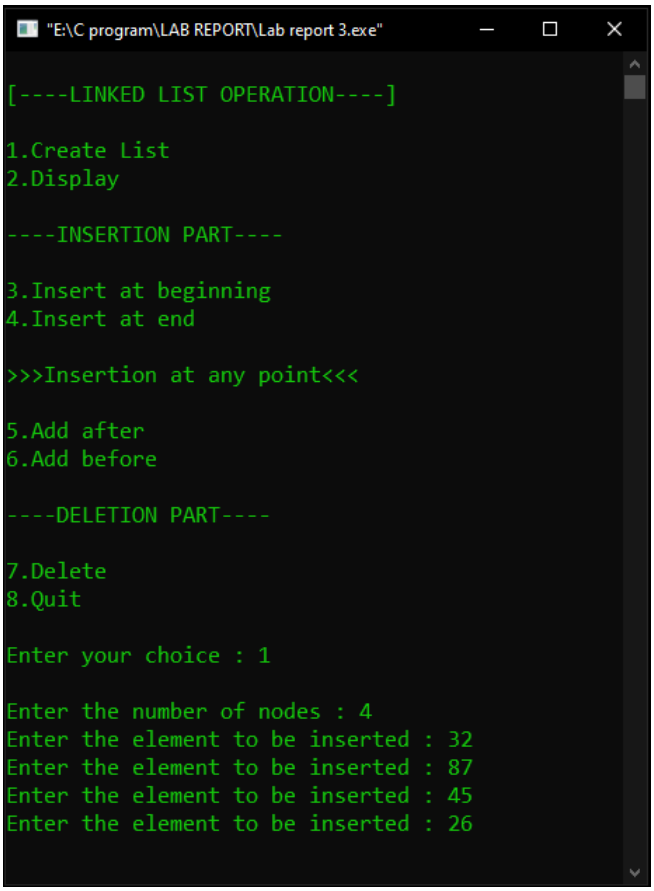
```
struct node *addtoempty(struct node *start,int data)
{
   struct node *tmp;
   tmp=(struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   tmp->prev=NULL;
   tmp->next=NULL;
   start=tmp;
   return start;
}

//insert at begainning function
struct node *addatbeg(struct node *start,int data)
{
   struct node *tmp;
   tmp = (struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   tmp->prev=NULL;
   tmp->next=start;
   start->prev=tmp;
   start=tmp;
   return start;
}

//insert at ending function
struct node *addatend(struct node *start,int data)
{
   struct node *tmp,*p;
   tmp=(struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   p=start;
   while(p->next!=NULL)
      p=p->next;
   p->next=tmp;
   tmp->next=NULL;
   tmp->prev=p;
   return start;
}

//Insertion At Any point
//add after function
struct node *addafter(struct node *start,int data,int item)
{
```

```
struct node *tmp,*p;
   tmp=(struct node *)malloc(sizeof(struct node));
   tmp->info=data;
   p=start;
   while(p!=NULL)
   {
      if(p->info==item)
      {
         tmp->prev=p;
         tmp->next=p->next;
         if(p->next!=NULL)
            p->next->prev=tmp;
         p->next=tmp;
         return start;
      }
      p=p->next;
   }
   printf("\n%d not present in the list\n\n",item);
   return start;
}

//add before function
struct node *addbefore(struct node *start,int data,int item)
{
   struct node *tmp,*q;
   if(start==NULL )
   {
      printf("\nList is empty\n");
      return start;
   }
   if(start->info==item)
   {
      tmp = (struct node *)malloc(sizeof(struct node));
      tmp->info=data;
      tmp->prev=NULL;
      tmp->next=start;
      start->prev=tmp;
      start=tmp;
      return start;
   }
   q=start;
   while(q!=NULL)
   {
```

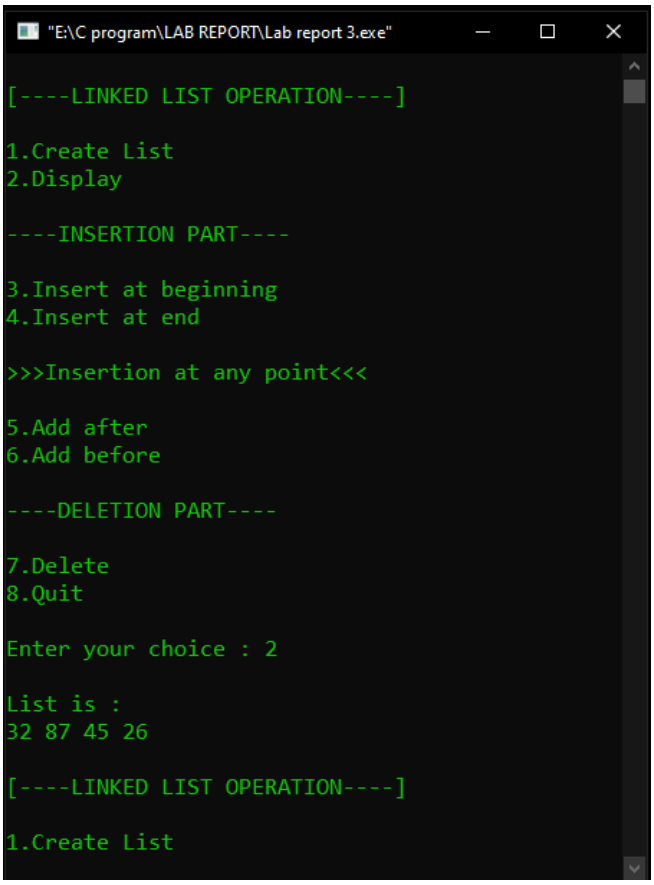
```
if(q->info==item)
        {
            tmp=(struct node *)malloc(sizeof(struct node));
            tmp->info=data;
            tmp->prev=q->prev;
            tmp->next = q;
            q->prev->next=tmp;
            q->prev=tmp;
            return start;
        }
        q=q->next;
    }
    printf("\n%d not present in the list\n",item);
    return start;
}

//Delete function
struct node *del(struct node *start,int data)
{
    struct node *tmp;
    if(start==NULL)
    {
        printf("\nList is empty\n");
        return start;
    }
    if(start->next==NULL)
        if(start->info==data)
        {
            tmp=start;
            start=NULL;
            free(tmp);
            return start;
        }
        else
        {
            printf("\nElement %d not found\n",data);
            return start;
        }
    /*Deletion of first node*/
    if(start->info==data)
    {
        tmp=start;
        start=start->next;
        start->prev=NULL;
```

```
free(tmp);
        return start;
    }
    /*Deletion in between*/
    tmp=start->next;
    while(tmp->next!=NULL )
    {
        if(tmp->info==data)
        {
            tmp->prev->next=tmp->next;
            tmp->next->prev=tmp->prev;
            free(tmp);
            return start;
        }
        tmp=tmp->next;
    }
    /*Deletion of last node*/
    if(tmp->info==data)
    {
        tmp->prev->next=NULL;
        free(tmp);
        return start;
    }
    printf("\nElement %d not found\n",data);
    return start;
}
```

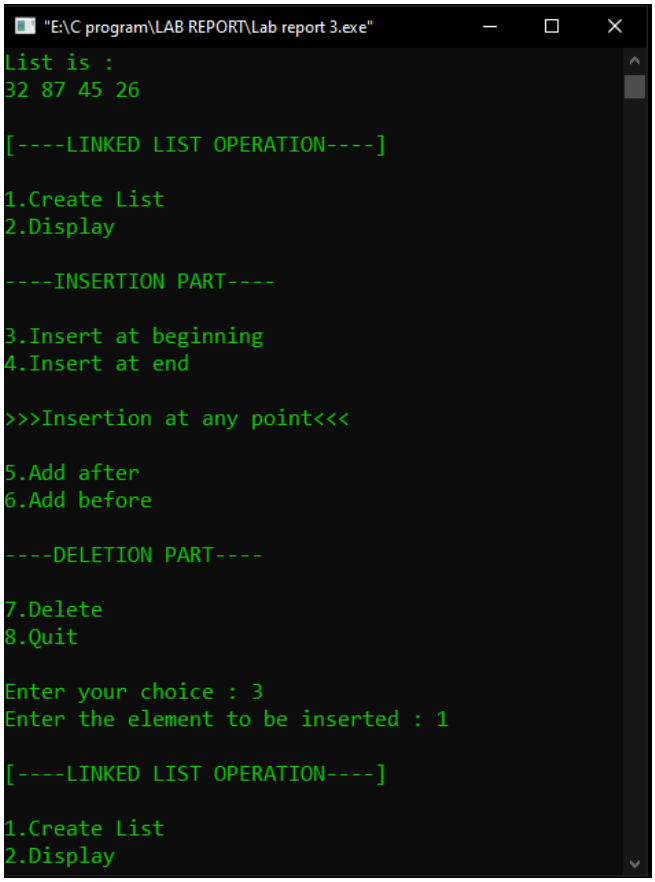
Output:



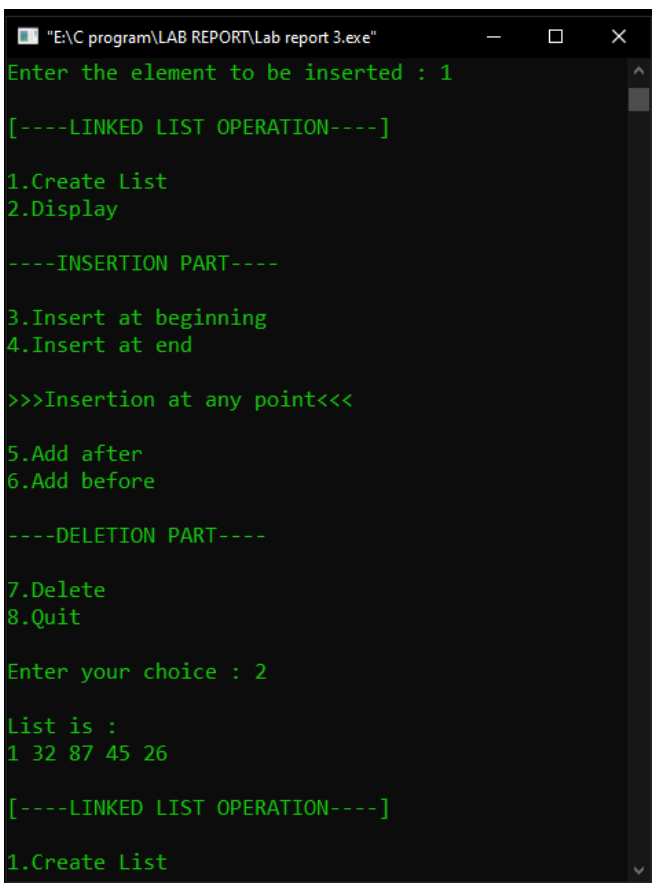
1.1 Create list



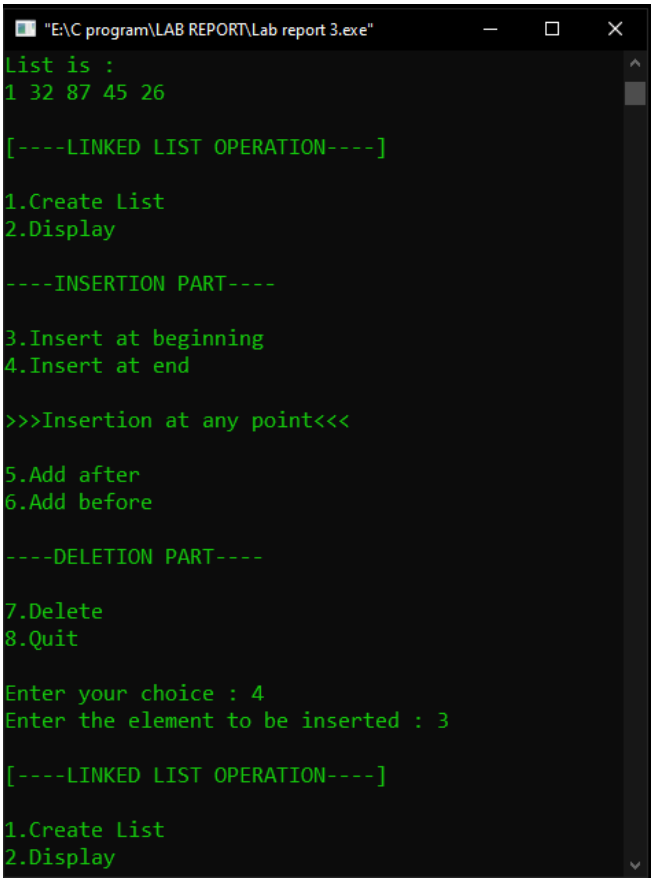
2.1 Display Created List



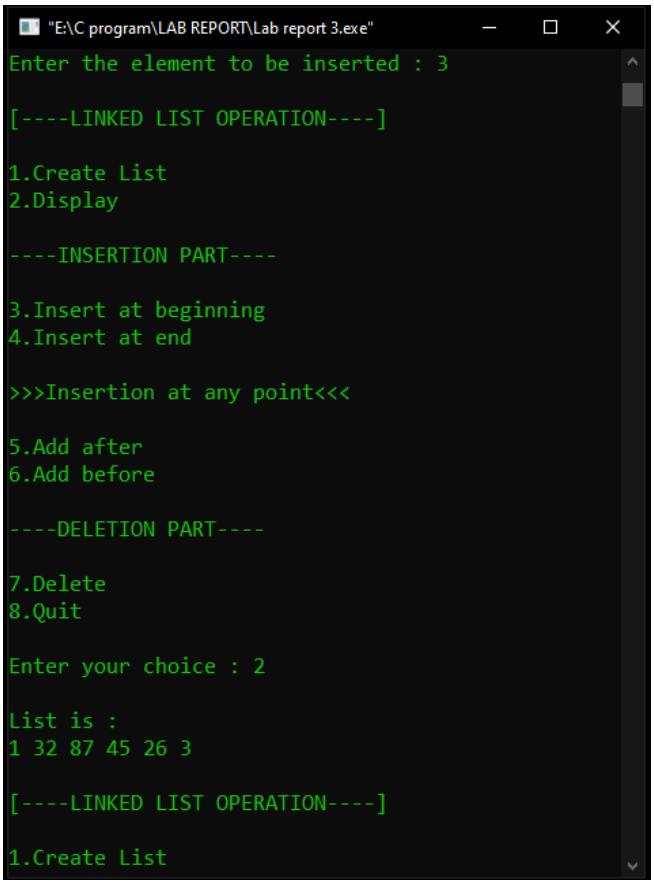
3.1 insert at beginning



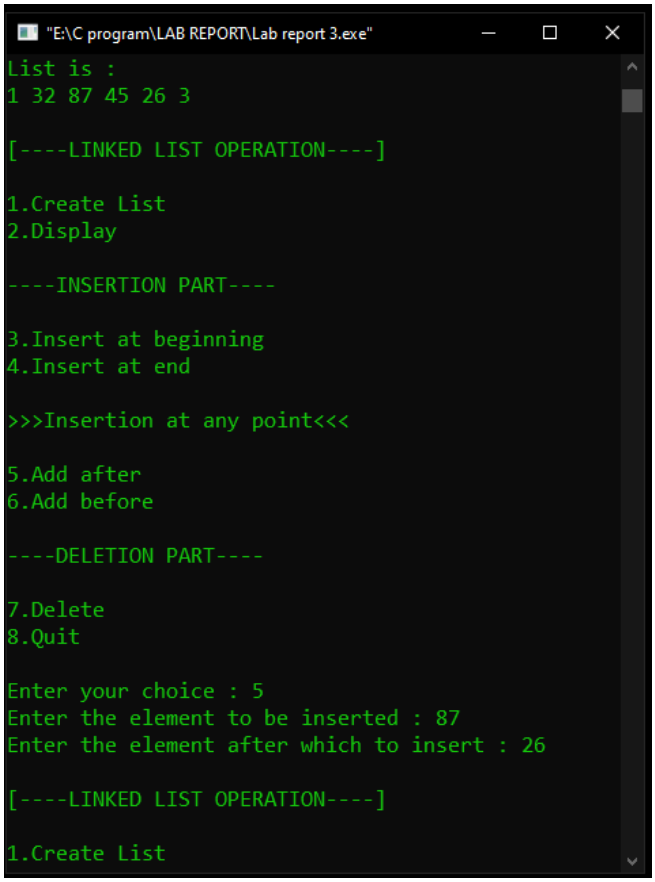
3.2 insert at begining display



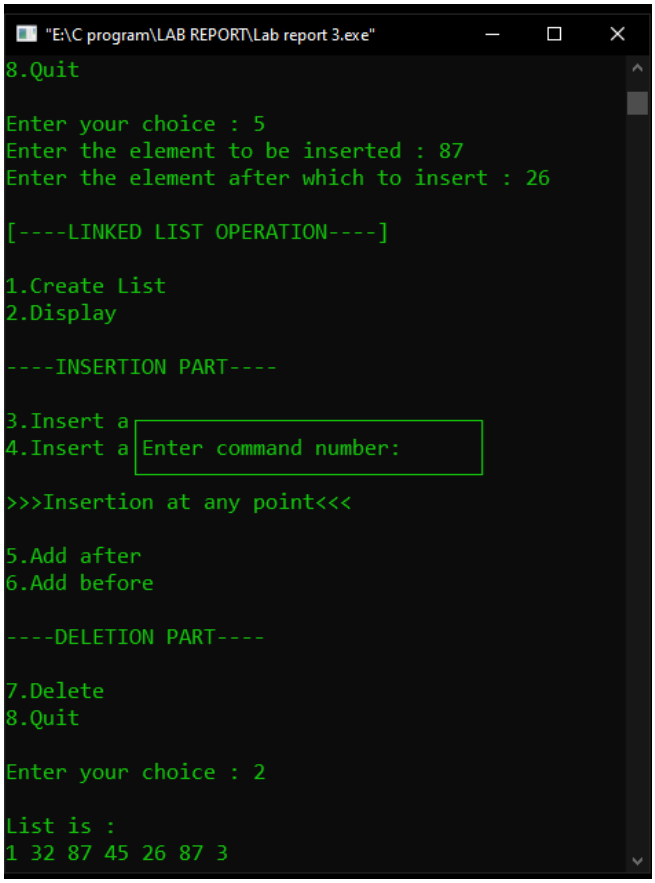
3.3 insert at ending



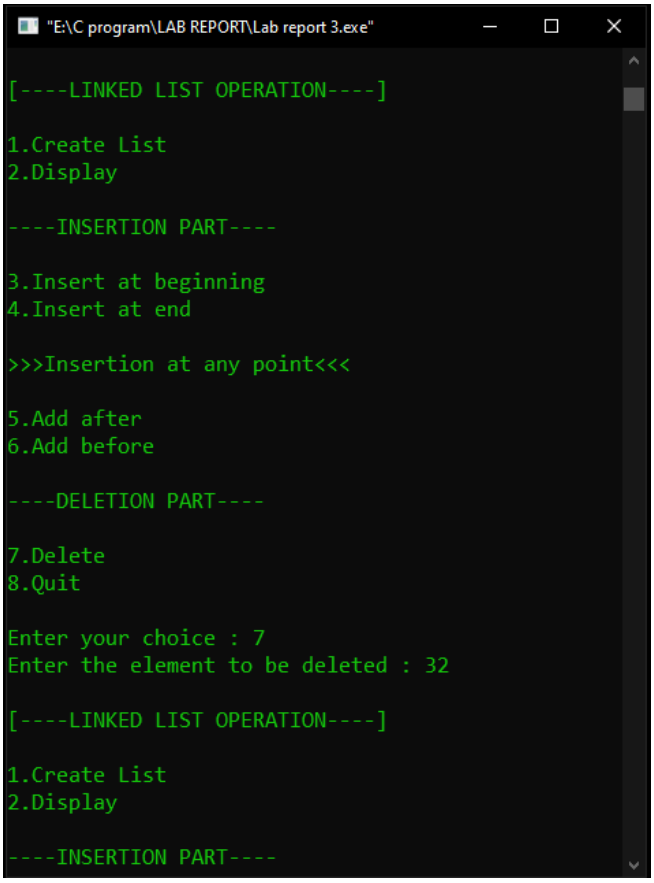
3.4 insert at ending display



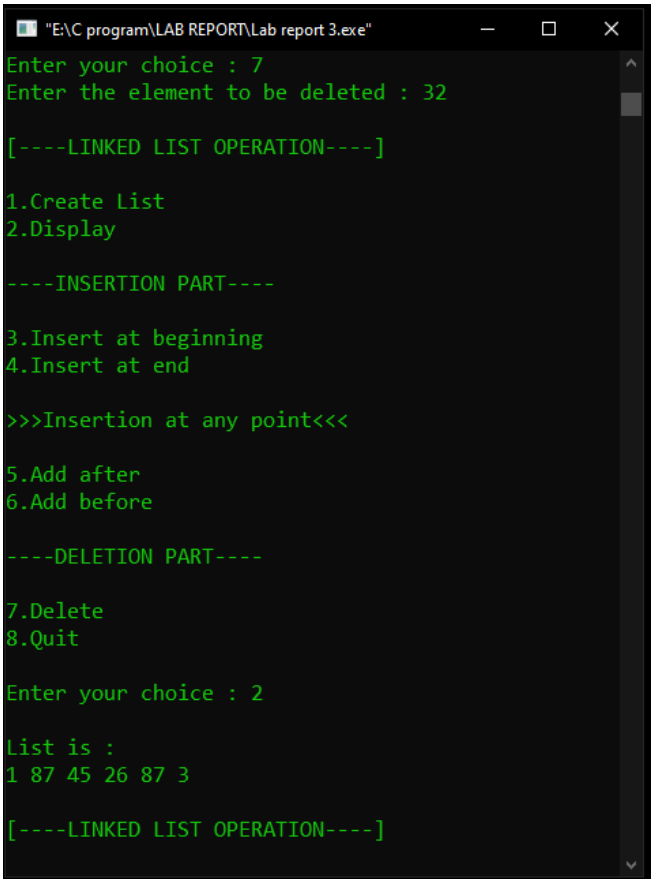
4.1 insert at any point



4.2 insert at any point display



5.1 delete perticular element



5.2 delete display

❑ Analysis and Discussion :

- We got the exact result on output. Sometimes the result was wrong but we found the right implementation.
- The problem of displaying anything in output is the easiest implementation. We solve that very easily.
- In this assignment, we faced some problems in this question but with the teacher's help we solve it.
- All program is easy to understand and these helped me a lot to remove my confusion about Linked list , array's operations, insertion ant any point , delation of node in linked list and many more operations of linked list .
- I learnt display something in program, Linked list operations as like node insertion and deletion , switch statement and application of user-defined function etc on program and many basic things about c programming.