PRACTICAL:- 03

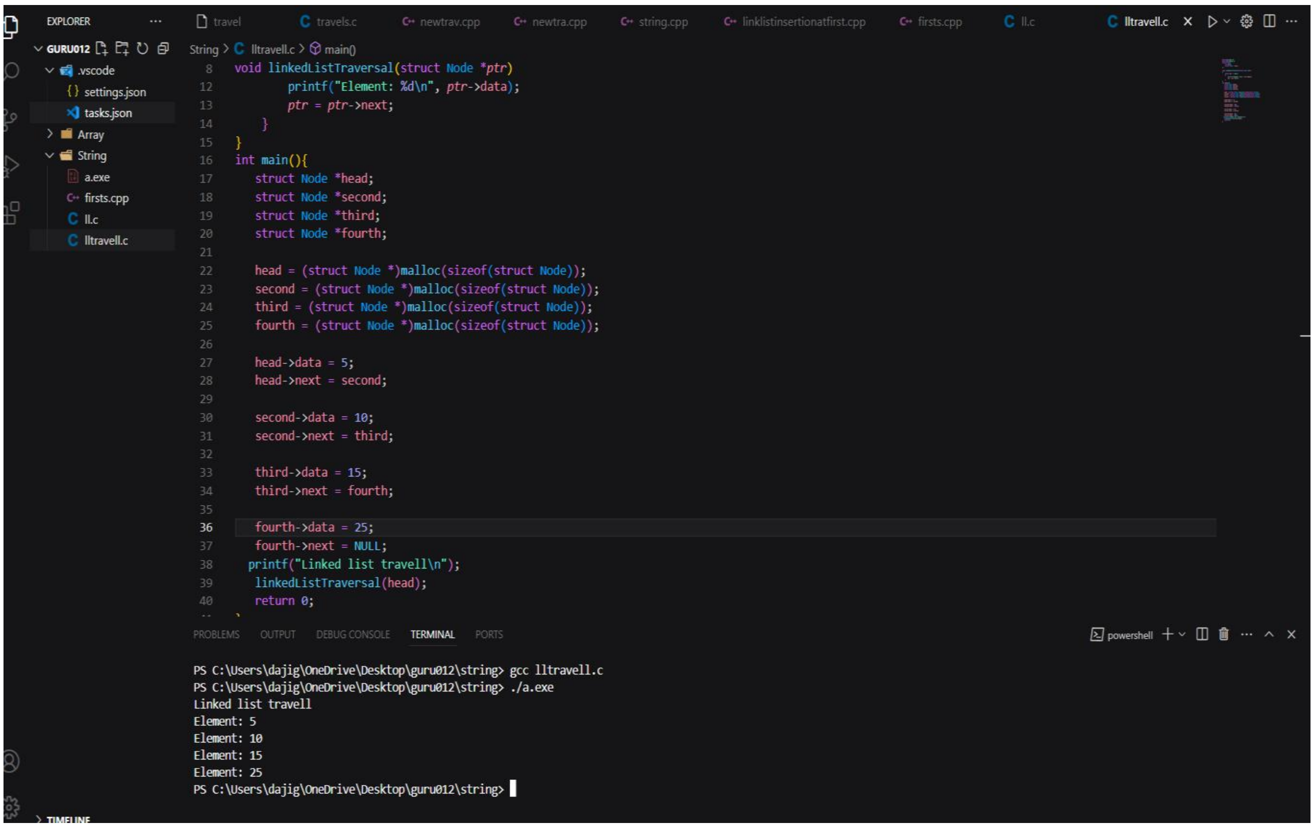
AIM:-[A] : Traversal in the link list.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
    int data;
    struct Node * next;
};

void linkedListTraversal(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}
int main(){
   struct Node *head;
   struct Node *second;
   struct Node *third;
   struct Node *fourth;

   head = (struct Node *)malloc(sizeof(struct Node));
   second = (struct Node *)malloc(sizeof(struct Node));
```

```
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 5;
    head->next = second;

    second->data = 10;
    second->next = third;

    third->data = 15;
    third->next = fourth;

    fourth->data = 25;
    fourth->next = NULL;
   printf("Linked list travell\n");
    linkedListTraversal(head);
    return 0;
}
```

[OUTPUT]



```
void linkedListTraversal(struct Node *ptr)
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}
int main(){
   struct Node *head;
   struct Node *second;
   struct Node *third;
   struct Node *fourth;

   head = (struct Node *)malloc(sizeof(struct Node));
   second = (struct Node *)malloc(sizeof(struct Node));
   third = (struct Node *)malloc(sizeof(struct Node));
   fourth = (struct Node *)malloc(sizeof(struct Node));

   head->data = 5;
   head->next = second;

   second->data = 10;
   second->next = third;

   third->data = 15;
   third->next = fourth;

   fourth->data = 25;
   fourth->next = NULL;
  printf("Linked list travell\n");
   linkedListTraversal(head);
   return 0;
```

```
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> gcc lltravell.c
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> ./a.exe
Linked list travell
Element: 5
Element: 10
Element: 15
Element: 25
PS C:\Users\dajig\OneDrive\Desktop\guru012\string>
```

[B]:- Inserting an element at beginning in linked list

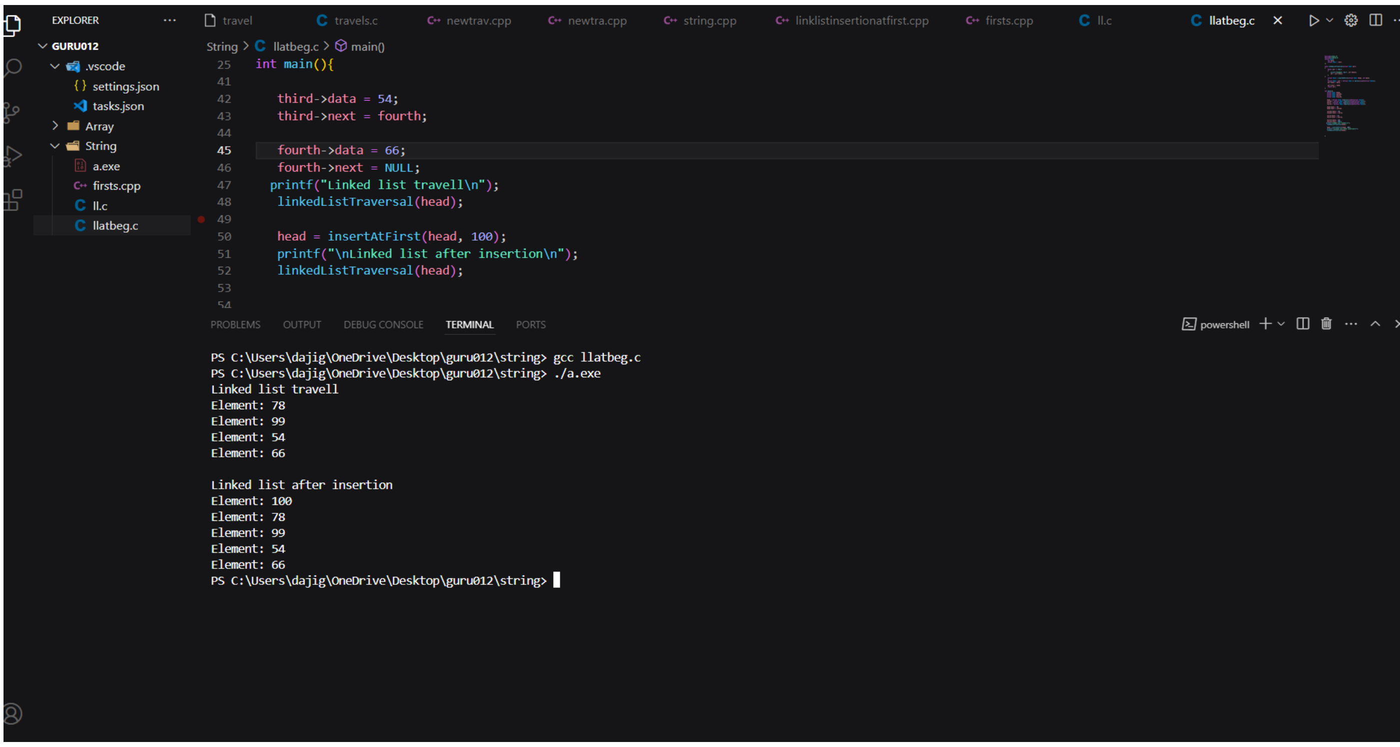
PROGARM:-

```
#include<stdio.h>

#include<stdlib.h>

struct Node{

   int data;

   struct Node * next;

};


void linkedListTraversal(struct Node *ptr)

{

   while (ptr != NULL)

   {

       printf("Element: %d\n", ptr->data);

       ptr = ptr->next;
```

```
    }
}
    struct Node * insertAtFirst(struct Node *head, int data)
    {
    struct Node * ptr = (struct Node *) malloc(sizeof(struct Node));
    ptr->data = data;

    ptr->next = head;
    return ptr;
}

int main(){
   struct Node *head;
   struct Node *second;
   struct Node *third;
   struct Node *fourth;

   head = (struct Node *)malloc(sizeof(struct Node));
   second = (struct Node *)malloc(sizeof(struct Node));
   third = (struct Node *)malloc(sizeof(struct Node));
   fourth = (struct Node *)malloc(sizeof(struct Node));

   head->data = 5;
   head->next = second;

   second->data = 10;
   second->next = third;

   third->data = 15;
   third->next = fourth;
```

```
  fourth->data = 25;

  fourth->next = NULL;

 printf("Linked list travell\n");

  linkedListTraversal(head);


  head = insertAtFirst(head, 100);

  printf("\nLinked list after insertion\n");

  linkedListTraversal(head);





}
```

[OUTPUT]



```
int main(){

    third->data = 54;
    third->next = fourth;

    fourth->data = 66;
    fourth->next = NULL;
   printf("Linked list travell\n");
    linkedListTraversal(head);

    head = insertAtFirst(head, 100);
    printf("\nLinked list after insertion\n");
    linkedListTraversal(head);
```

```
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> gcc llatbeg.c
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> ./a.exe
Linked list travell
Element: 78
Element: 99
Element: 54
Element: 66

Linked list after insertion
Element: 100
Element: 78
Element: 99
Element: 54
Element: 66
PS C:\Users\dajig\OneDrive\Desktop\guru012\string>
```

[C]:- Inserting element at specific index or position in linked list.

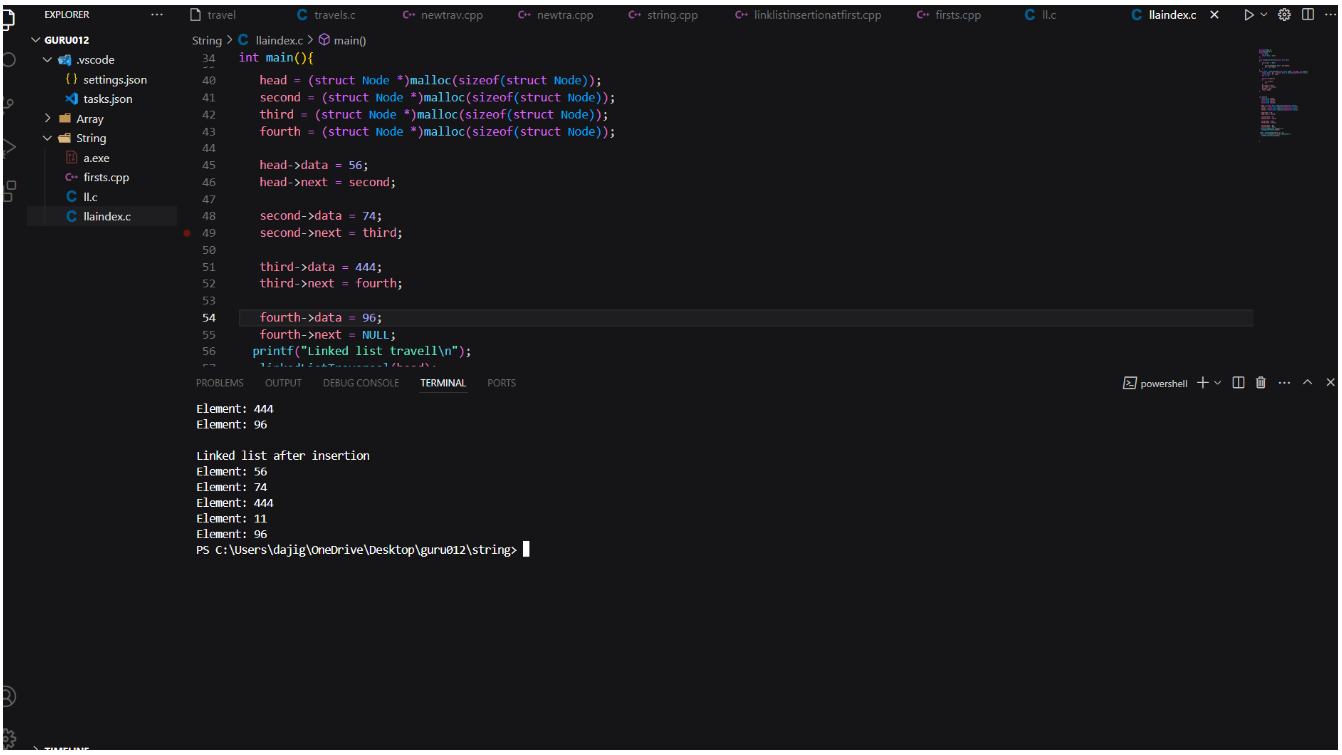
PROGARM:- #include<stdio.h>

```
#include<stdlib.h>
struct Node{
   int data;
   struct Node * next;
};

void linkedListTraversal(struct Node *ptr)
{
   while (ptr != NULL)
   {
      printf("Element: %d\n", ptr->data);
      ptr = ptr->next;
   }
}
struct Node * insertAtIndex(struct Node *head, int data, int index){
   struct Node * ptr = (struct Node *) malloc(sizeof(struct Node));
   struct Node * p = head;
   int i = 0;

   while (i!=index-1)
   {
      p = p->next;
      i++;
   }
   ptr->data = data;
   ptr->next = p->next;
   p->next = ptr;
   return head;
}
```

```
int main(){
   struct Node *head;
   struct Node *second;
   struct Node *third;
   struct Node *fourth;

   head = (struct Node *)malloc(sizeof(struct Node));
   second = (struct Node *)malloc(sizeof(struct Node));
   third = (struct Node *)malloc(sizeof(struct Node));
   fourth = (struct Node *)malloc(sizeof(struct Node));

   head->data = 56;
   head->next = second;

   second->data = 74;
   second->next = third;

   third->data = 444;
   third->next = fourth;

   fourth->data = 96;
   fourth->next = NULL;
  printf("Linked list travell\n");
   linkedListTraversal(head);

 head = insertAtIndex(head, 11, 3);
   printf("\nLinked list after insertion\n");
   linkedListTraversal(head);
```

}

[OUTPUT]



```c
int main(){
    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 56;
    head->next = second;

    second->data = 74;
    second->next = third;

    third->data = 444;
    third->next = fourth;

    fourth->data = 96;
    fourth->next = NULL;
  printf("Linked list travell\n");
```

```
Element: 444
Element: 96

Linked list after insertion
Element: 56
Element: 74
Element: 444
Element: 11
Element: 96
PS C:\Users\dajig\OneDrive\Desktop\guru012\string>
```

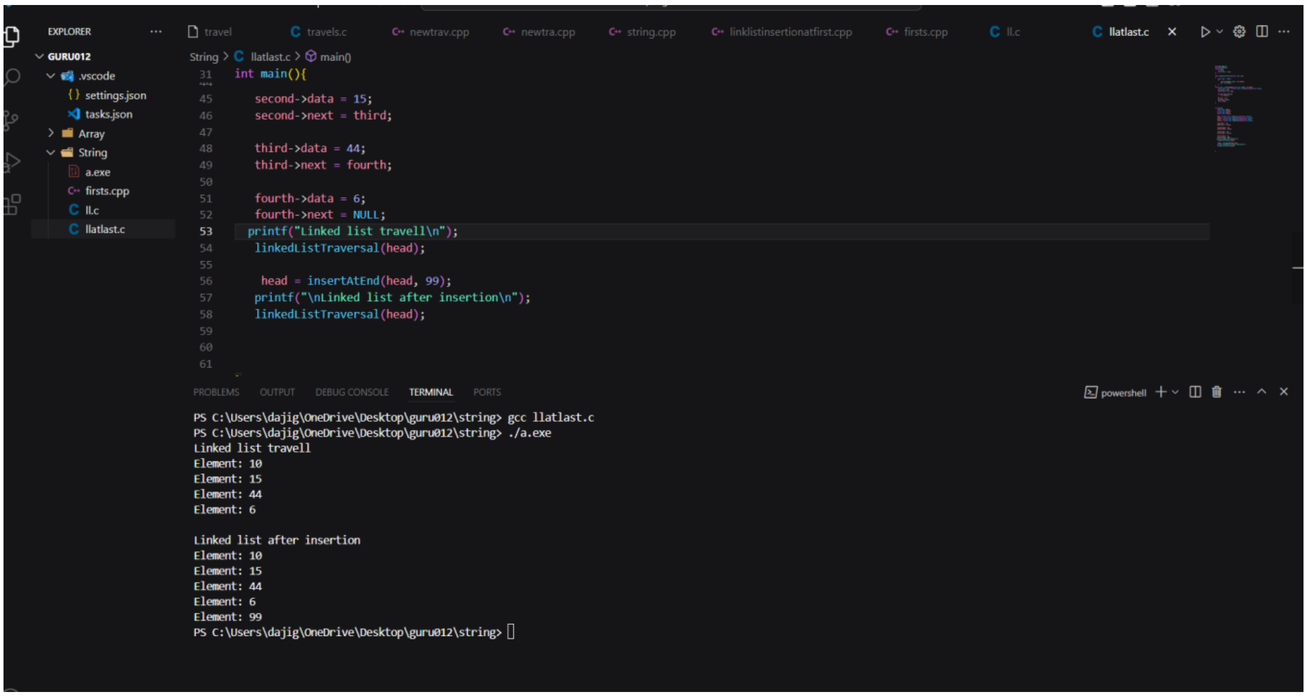
Insertion element at last position in linked list.

PROGRAM:-

```c
#include<stdio.h>

#include<stdlib.h>

struct Node{

    int data;

    struct Node * next;

};


void linkedListTraversal(struct Node *ptr)

{

    while (ptr != NULL)

    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}
struct Node * insertAtEnd(struct Node *head, int data){
    struct Node * ptr = (struct Node *) malloc(sizeof(struct Node));
    ptr->data = data;
    struct Node * p = head;

    while(p->next!=NULL){
        p = p->next;
    }
    p->next = ptr;
    ptr->next = NULL;
    return head;
}


int main(){
   struct Node *head;
   struct Node *second;
   struct Node *third;
   struct Node *fourth;

   head = (struct Node *)malloc(sizeof(struct Node));
   second = (struct Node *)malloc(sizeof(struct Node));
   third = (struct Node *)malloc(sizeof(struct Node));
   fourth = (struct Node *)malloc(sizeof(struct Node));

   head->data = 10;
   head->next = second;
```

```
  second->data = 15;

  second->next = third;


  third->data = 44;

  third->next = fourth;


  fourth->data = 6;

  fourth->next = NULL;

 printf("Linked list travell\n");

  linkedListTraversal(head);


   head = insertAtEnd(head, 99);

  printf("\nLinked list after insertion\n");

  linkedListTraversal(head);




}
```

[OUTPUT]



```
int main(){

   second->data = 15;
   second->next = third;

   third->data = 44;
   third->next = fourth;

   fourth->data = 6;
   fourth->next = NULL;
 printf("Linked list travell\n");
   linkedListTraversal(head);

    head = insertAtEnd(head, 99);
   printf("\nLinked list after insertion\n");
   linkedListTraversal(head);
```

```
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> gcc llatlast.c
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> ./a.exe
Linked list travell
Element: 10
Element: 15
Element: 44
Element: 6

Linked list after insertion
Element: 10
Element: 15
Element: 44
Element: 6
Element: 99
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> 
```

DELETATION PROGRAMS

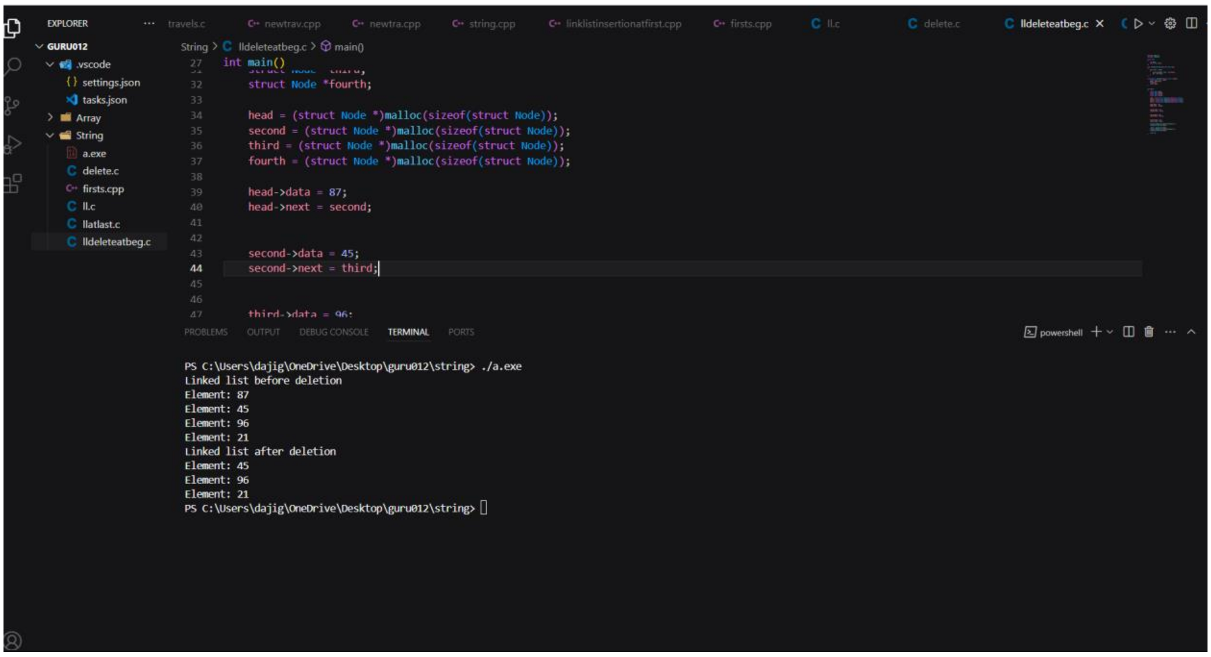
Delete element at beginning in linked list.

PROGRAM:-

```
#include <stdio.h>

#include <stdlib.h>



struct Node

{

   int data;

   struct Node *next;

};



void linkedListTraversal(struct Node *ptr)

{

   while (ptr != NULL)

   {

      printf("Element: %d\n", ptr->data);
```

```
        ptr = ptr->next;
    }
}

struct Node * deleteFirst(struct Node * head){
    struct Node * ptr = head;
    head = head->next;
    free(ptr);
    return head;
}

int main()
{
    struct Node *head;
    struct Node *second;
    struct Node *third;
    struct Node *fourth;

    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 87;
    head->next = second;


    second->data = 45;
    second->next = third;
    third->data = 96;
    third->next = fourth;



    fourth->data = 21;
    fourth->next = NULL;

    printf("Linked list before deletion\n");
    linkedListTraversal(head);

     head = deleteFirst(head);
    printf("Linked list after deletion\n");
    linkedListTraversal(head);

    return 0;
}
```

[OUTPUT]



```
int main()
    struct Node *fourth;

    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 87;
    head->next = second;


    second->data = 45;
    second->next = third;


    third->data = 96;
```

```
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> ./a.exe
Linked list before deletion
Element: 87
Element: 45
Element: 96
Element: 21
Linked list after deletion
Element: 45
Element: 96
Element: 21
PS C:\Users\dajig\OneDrive\Desktop\guru012\string>
```

Delete element at specific index in linked list.

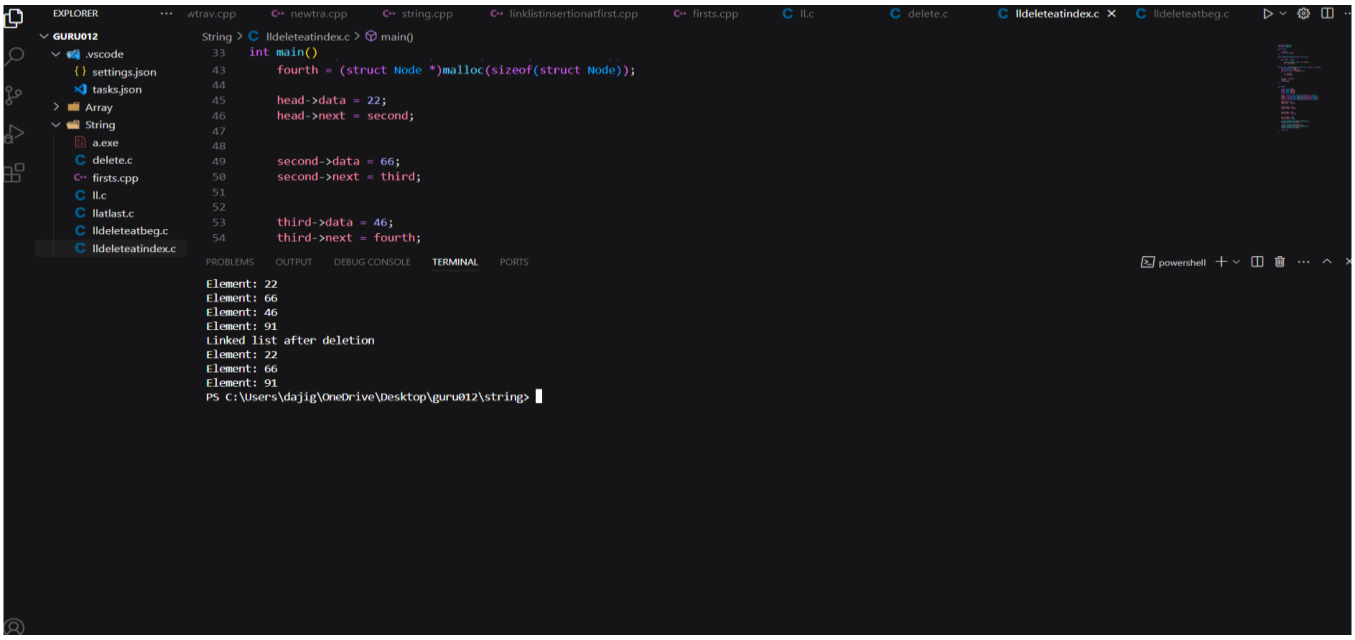
PROGRAM:-

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

struct Node
{
   int data;
   struct Node *next;
};

void linkedListTraversal(struct Node *ptr)
{
   while (ptr != NULL)
   {
      printf("Element: %d\n", ptr->data);
      ptr = ptr->next;
   }
}
struct Node * deleteAtIndex(struct Node * head, int index){
   struct Node *p = head;
   struct Node *q = head->next;
   for (int i = 0; i < index-1; i++)
   {
      p = p->next;
      q = q->next;
   }

   p->next = q->next;
   free(q);
   return head;
}
```

```
int main()
{
  struct Node *head;
  struct Node *second;
  struct Node *third;
  struct Node *fourth;

  head = (struct Node *)malloc(sizeof(struct Node));
  second = (struct Node *)malloc(sizeof(struct Node));
  third = (struct Node *)malloc(sizeof(struct Node));
  fourth = (struct Node *)malloc(sizeof(struct Node));

  head->data = 22;
  head->next = second;



  second->data = 66;
  second->next = third;



  third->data = 46;
  third->next = fourth;



  fourth->data = 91;
  fourth->next = NULL;

  printf("Linked list before deletion\n");
  linkedListTraversal(head);
```

```
     head = deleteAtIndex(head, 2);
    printf("Linked list after deletion\n");
    linkedListTraversal(head);


    return 0;
}
```

[OUTPUT]



```
int main()
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 22;
    head->next = second;


    second->data = 66;
    second->next = third;


    third->data = 46;
    third->next = fourth;
```

```
Element: 22
Element: 66
Element: 46
Element: 91
Linked list after deletion
Element: 22
Element: 66
Element: 91
PS C:\Users\dajig\OneDrive\Desktop\guru012\string>
```

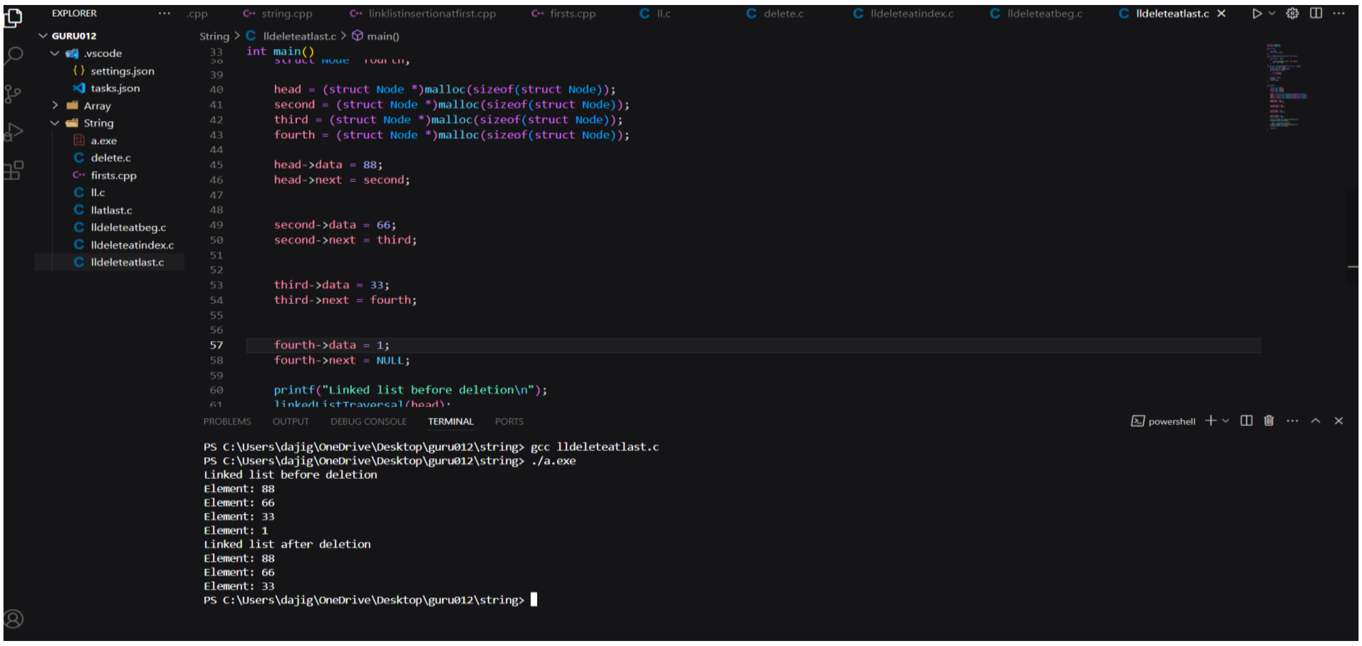
Delete element at last in linked list.

PROGARM:-

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

struct Node
{
    int data;
    struct Node *next;
};
```

```
void linkedListTraversal(struct Node *ptr)
{
    while (ptr != NULL)
    {
        printf("Element: %d\n", ptr->data);
        ptr = ptr->next;
    }
}
struct Node * deleteAtLast(struct Node * head){
    struct Node *p = head;
    struct Node *q = head->next;
    while(q->next !=NULL)
    {
        p = p->next;
        q = q->next;
    }

    p->next = NULL;
    free(q);
    return head;
}

int main()
{
    struct Node *head;
    struct Node *second;
    struct Node *third;
    struct Node *fourth;

    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 88;
    head->next = second;



    second->data = 66;
    second->next = third;



    third->data = 33;
    third->next = fourth;



    fourth->data = 1;
    fourth->next = NULL;

    printf("Linked list before deletion\n");
    linkedListTraversal(head);

      head = deleteAtLast(head);
    printf("Linked list after deletion\n");
    linkedListTraversal(head);

    return 0;
}
```

[OUTPUT]



```c
int main()
    struct Node *fourth;

    head = (struct Node *)malloc(sizeof(struct Node));
    second = (struct Node *)malloc(sizeof(struct Node));
    third = (struct Node *)malloc(sizeof(struct Node));
    fourth = (struct Node *)malloc(sizeof(struct Node));

    head->data = 88;
    head->next = second;


    second->data = 66;
    second->next = third;


    third->data = 33;
    third->next = fourth;


    fourth->data = 1;
    fourth->next = NULL;

    printf("Linked list before deletion\n");
    linkedListTraversal(head);
```

```
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> gcc lldeleteatlast.c
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> ./a.exe
Linked list before deletion
Element: 88
Element: 66
Element: 33
Element: 1
Linked list after deletion
Element: 88
Element: 66
Element: 33
PS C:\Users\dajig\OneDrive\Desktop\guru012\string>
```

Github Link: https://github.com/guru24961/Data-Stracture-practical.git