

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]

```
String > C lltravell.c > main()
8 void linkedListTraversal(struct Node *ptr)
12     printf("Element: %d\n", ptr->data);
13     ptr = ptr->next;
14 }
15
16 int main(){
17     struct Node *head;
18     struct Node *second;
19     struct Node *third;
20     struct Node *fourth;
21
22     head = (struct Node *)malloc(sizeof(struct Node));
23     second = (struct Node *)malloc(sizeof(struct Node));
24     third = (struct Node *)malloc(sizeof(struct Node));
25     fourth = (struct Node *)malloc(sizeof(struct Node));
26
27     head->data = 5;
28     head->next = second;
29
30     second->data = 10;
31     second->next = third;
32
33     third->data = 15;
34     third->next = fourth;
35
36     fourth->data = 25;
37     fourth->next = NULL;
38     printf("Linked list travell\n");
39     linkedListTraversal(head);
40     return 0;
41 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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

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 * 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]

The screenshot shows a Visual Studio Code editor with a C++ project named 'travel'. The file explorer on the left shows the project structure, including a 'String' folder with files like 'a.exe', 'firsts.cpp', 'll.c', and 'llatbeg.c'. The main editor window displays the source code for 'llatbeg.c', which includes the implementation of a linked list with functions for traversal and insertion. The terminal at the bottom shows the execution of the program, displaying the output of the linked list traversal and insertion operations.

```

String > C:\llatbeg.c> main()
25  int main(){
41
42      third->data = 54;
43      third->next = fourth;
44
45      fourth->data = 66;
46      fourth->next = NULL;
47      printf("Linked list travell\n");
48      linkedListTraversal(head);
49
50      head = insertAtFirst(head, 100);
51      printf("\nLinked list after insertion\n");
52      linkedListTraversal(head);
53
54
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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]

```

// llindex.c
#include <stdio.h>
#include <stdlib.h>

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

int main() {
    struct Node *head, *second, *third, *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 traversal\n");

    // Traversal
    struct Node *temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

```

Output:

```

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:-

```
#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]

```
String > C llatlast.c > main()
31 int main()
45     second->data = 15;
46     second->next = third;
47
48     third->data = 44;
49     third->next = fourth;
50
51     fourth->data = 6;
52     fourth->next = NULL;
53     printf("Linked list traversal\n");
54     linkedListTraversal(head);
55
56     head = insertAtEnd(head, 99);
57     printf("\nLinked list after insertion\n");
58     linkedListTraversal(head);
59
60
61
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> gcc llatlast.c
PS C:\Users\dajig\OneDrive\Desktop\guru012\string> ./a.exe
Linked list traversal
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);
```

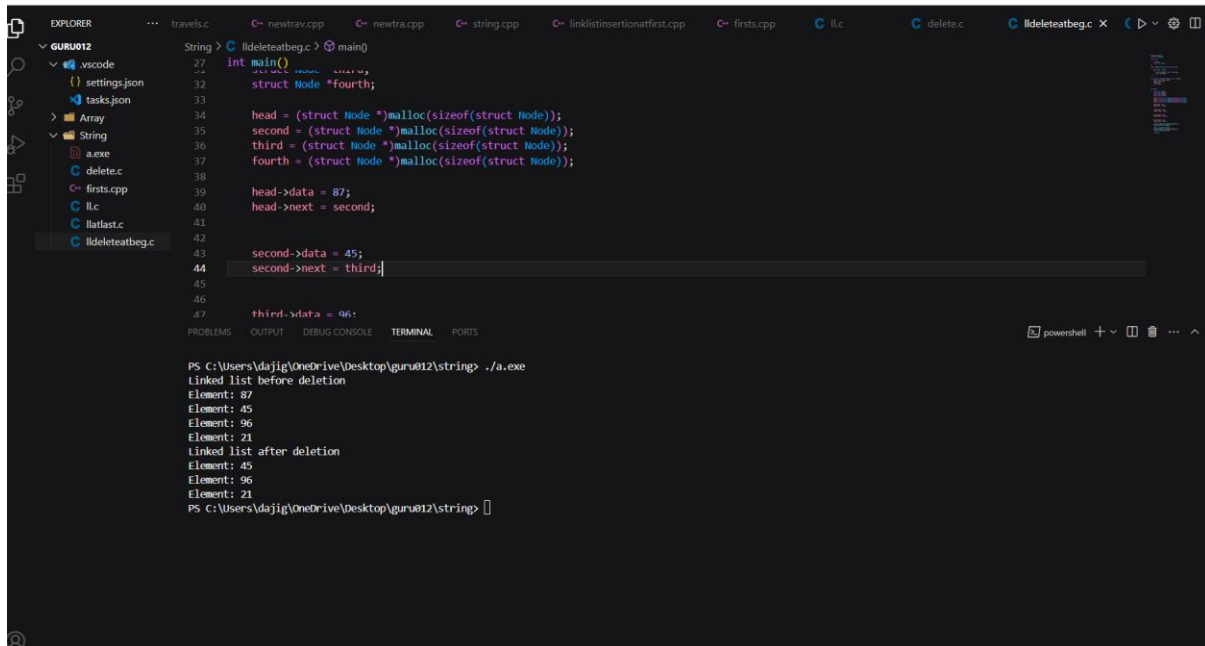
```
printf("Linked list after deletion\n");
```

```
linkedListTraversal(head);
```

```
return 0;
```

```
}
```

[OUTPUT]



```
String > C ldeleteatbeg.c > main()
27 int main()
32     struct Node *fourth;
33
34     head = (struct Node *)malloc(sizeof(struct Node));
35     second = (struct Node *)malloc(sizeof(struct Node));
36     third = (struct Node *)malloc(sizeof(struct Node));
37     fourth = (struct Node *)malloc(sizeof(struct Node));
38
39     head->data = 87;
40     head->next = second;
41
42
43     second->data = 45;
44     second->next = third;
45
46
47     third->data = 96;
48
49
50     fourth->data = 21;
51     fourth->next = NULL;
52
53     printf("Linked list before deletion\n");
54     linkedListTraversal(head);
55
56     head = deleteFirst(head);
57
58     printf("Linked list after deletion\n");
59     linkedListTraversal(head);
60
61     return 0;
62 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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]

```

String > C ldeleteatindex.c > main()
33 int main()
43 fourth = (struct Node *)malloc(sizeof(struct Node));
44
45 head->data = 22;
46 head->next = second;
47
48
49 second->data = 66;
50 second->next = third;
51
52
53 third->data = 46;
54 third->next = fourth;

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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]

```
String > lldeleteatlast.c > main()
33 int main()
34 {
35     struct Node *head = NULL;
36
37     head = (struct Node *)malloc(sizeof(struct Node));
38     second = (struct Node *)malloc(sizeof(struct Node));
39     third = (struct Node *)malloc(sizeof(struct Node));
40     fourth = (struct Node *)malloc(sizeof(struct Node));
41
42     head->data = 88;
43     head->next = second;
44
45     second->data = 66;
46     second->next = third;
47
48     third->data = 33;
49     third->next = fourth;
50
51     fourth->data = 1;
52     fourth->next = NULL;
53
54     printf("Linked list before deletion\n");
55     linkedlisttraversal(head);
56
57     return 0;
58 }
```

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>