



```
GNU nano 4.8                               LinkedList.c
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int num;
    struct node *nextptr;
} *stnode;

void createNodeList(int n)
{
    struct node *fnNode, *tmp;
    int num, i;
    stnode = (struct node *)malloc(sizeof(struct node));

    if(stnode == NULL)
    {
        printf(" Memory can not be allocated.");
    }
    else
    {

        printf(" Input data for node 1 : ");
        scanf("%d", &num);
        stnode->num = num;
        stnode->nextptr = NULL;
        tmp = stnode;

        for(i=2; i<=n; i++)
        {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode == NULL)
            {
                printf(" Memory can not be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);

                fnNode->num = num;
                fnNode->nextptr = NULL;

                tmp->nextptr = fnNode;
                tmp = tmp->nextptr;
            }
        }
    }
}
```

Read 84 lines

Get Help Write Out Where Is Cut Text Justify Cur Pos M-U Undo M-A Mark Text M-] To Bracket M-0 Previous ^B Back ^\_ Prev Word

26 Aug 2020 (Lab 3)

```
rishabh@DESKTOP-AUG0508U: /media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020
GNU nano 4.8                               LinkedList.c                               Modified

fnNode = (struct node *)malloc(sizeof(struct node));
if(fnNode == NULL)
{
    printf(" Memory can not be allocated.");
    break;
}
else
{
    printf(" Input data for node %d : ", i);
    scanf(" %d", &num);

    fnNode->num = num;
    fnNode->nextptr = NULL;

    tmp->nextptr = fnNode;
    tmp = tmp->nextptr;
}
}
}
}
}
void displayList()
{
    struct node *tmp;
    if(stnode == NULL)
    {
        printf(" List is empty.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);
            tmp = tmp->nextptr;
        }
    }
}

void main()
{
    int n;
    printf("\n\n Linked List : To create and display Singly Linked List :\n");
    printf("-----\n");

    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    createNodeList(n);
    printf("\n Data entered in the list : \n");
    displayList();
}
```

**Footer:**

<b>^G</b> Get Help	<b>^O</b> Write Out	<b>^W</b> Where Is	<b>^K</b> Cut Text	<b>^J</b> Justify	<b>^C</b> Cur Pos	<b>M-U</b> Undo	<b>M-A</b> Mark Text	<b>M-]</b> To Bracket	<b>M-Q</b> Previous	<b>^B</b> Back	<b>^_</b> Prev Word
<b>^X</b> Exit	<b>^R</b> Read File	<b>^_</b> Replace	<b>^U</b> Paste Text	<b>^T</b> To Spell	<b>^</b> Go To Line	<b>M-E</b> Redo	<b>M-6</b> Copy Text	<b>^O</b> Where Was	<b>M-W</b> Next	<b>^F</b> Forward	<b>^_</b> Next Word

26 Aug 2020 (Lab 3)

```
rishabh@DESKTOP-AUG0508U: /media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020
rishabh@DESKTOP-AUG0508U:/media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020$ nano LinkedList.c
rishabh@DESKTOP-AUG0508U:/media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020$ gcc LinkedList.c
rishabh@DESKTOP-AUG0508U:/media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020$ ls
a.out  LinkedList.c
rishabh@DESKTOP-AUG0508U:/media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020$ ./a.out

Linked List : To create and display Singly Linked List :
-----
Input the number of nodes : 6
Input data for node 1 : 7
Input data for node 2 : 7
Input data for node 3 : 4
Input data for node 4 : 9
Input data for node 5 : 1
Input data for node 6 : 3

Data entered in the list :
Data = 7
Data = 7
Data = 4
Data = 9
Data = 1
Data = 3
rishabh@DESKTOP-AUG0508U:/media/rishabh/Backup Plus/Vlabs/DS Labs/Lab Aug 26 2020$
```

## Q2 ) What is the purpose of using Linked List for problem solving ?

### **Dynamic Data Structure**

> Linked list is a dynamic data structure so it can grow and shrink at runtime by allocating and deallocating memory. So there is no need to give initial size of the linked list.

### **Insertion and Deletion**

> Unlike array here we don't have to shift elements after insertion or deletion of the element. In linked list we just have to update the address of next pointer of a node.

### **No Memory Wastage**

> As size of linked list can increase or decrease at run time so there is no memory wastage. In array there is lot of memory wastage, like if we declare an array of size 10 and store only 6 elements in it then space of 4 elements are wasted. There is no such issue in linked list as memory is allocated only and when required.

### Q3) What data type will you use for the elements in a linked list ? (Show an Example)

In linked list, we don't have pointer data type, that links to address of the next node which may or may not contain same data type.

In Linked List, we just need valid pointers. For Example : -

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