```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<ctime>
#include<string>
using namespace std;

/*Node Declaration*/

double t1,t2;


struct node

{

    int info;
    struct node *next;

}*initialize;



/*Class Declaration*/

class single_llist

{
    public:

        node* create_node(int);
        void insert_begin();
        void insert_pos();
        void insert_last();
        void delete_pos();
        void search();
        void count();
        void display();

        single_llist()
        {

            initialize = NULL;
            cout<<"Constructor is called\n"<<endl;

        }


};


/*Node creation*/

node *single_llist::create_node(int value)
```

```
{

    struct node *temp, *s;
       temp = new(struct node);
       if (temp == NULL)

    {

        cout<<"Memory not allocated "<<endl;
              return 0;

    }

    else

    {

        temp->info = value;

        temp->next = NULL;

        return temp;

    }

}



/*Inserting element in beginning*/

void single_llist::insert_begin()

{
    int value;

cout<<"Enter number of nodes"<<endl;

cin>>value;

while(value!=0)
{

    struct node *temp, *p;

    temp = create_node(rand()%10000);

    if (initialize == NULL)

    {

        initialize = temp;
```

```
        initialize->next = NULL;

    }

    else

    {
       p = initialize;
        initialize = temp;
        initialize->next = p;

    }
   value--;
}


}



/*Inserting Node at last*/

void single_llist::insert_last()

{

    int value;

    cout<<"Enter the value to be inserted: ";

    cin>>value;

    struct node *temp, *s;

    temp = create_node(value);

    s = initialize;

    while (s->next != NULL)

    {

        s = s->next;

    }

    temp->next = NULL;

    s->next = temp;

    cout<<"Element Inserted at last"<<endl;

}
```

```
/*Insertion of node at a given position*/
void single_llist::insert_pos()
{
    int value, pos, counter = 0;
    cout<<"Enter the value to be inserted: ";
    cin>>value;
    struct node *temp, *s, *ptr;
    temp = create_node(value);
    cout<<"Enter the postion at which node to be inserted: ";
    cin>>pos;
    int i;
    s = initialize;
    while (s != NULL)
    {
        s = s->next;
        counter++;
    }
    if (pos == 1)
    {
        if (initialize == NULL)
        {
            initialize = temp;
            initialize->next = NULL;
        }
        else
        {
```

```
            ptr = initialize;
            initialize = temp;
            initialize->next = ptr;
        }
    }
    else if (pos > 1  && pos <= counter)
    {
        s = initialize;
        for (i = 1; i < pos; i++)
        {
            ptr = s;
            s = s->next;
        }
        ptr->next = temp;
        temp->next = s;
    }
    else
    {
        cout<<"Positon out of range"<<endl;
    }
}

/*Delete element at a given position*/
void single_llist::delete_pos()
{
    int pos, i, counter = 0;
    if (initialize == NULL)
```

```
{
    cout<<"List is empty"<<endl;
    return;
}
cout<<"Enter the position of value to be deleted: ";
cin>>pos;
struct node *s, *ptr;
s = initialize;
if (pos == 1)
{
    initialize = s->next;
}
else
{
    while (s != NULL)
    {
                s = s->next;
        counter++;
    }
    if (pos > 0 && pos <= counter)
    {
        s = initialize;
        for (i = 1;i < pos;i++)
        {
            ptr = s;
            s = s->next;
        }
        ptr->next = s->next;
```

```
        }
        else
        {
            cout<<"Position out of range"<<endl;
        }
        free(s);
        cout<<"Element Deleted"<<endl;
    }
}
/*Searching an element*/
void single_llist::search()
{
    int value, pos = 0;
    bool flag = false;
    if (initialize == NULL)
    {
        cout<<"List is empty"<<endl;
        return;
    }
    cout<<"Enter the value to be searched: ";
    cin>>value;
    struct node *s;
    s = initialize;
    while (s != NULL)
    {
        pos++;
        if (s->info == value)
```

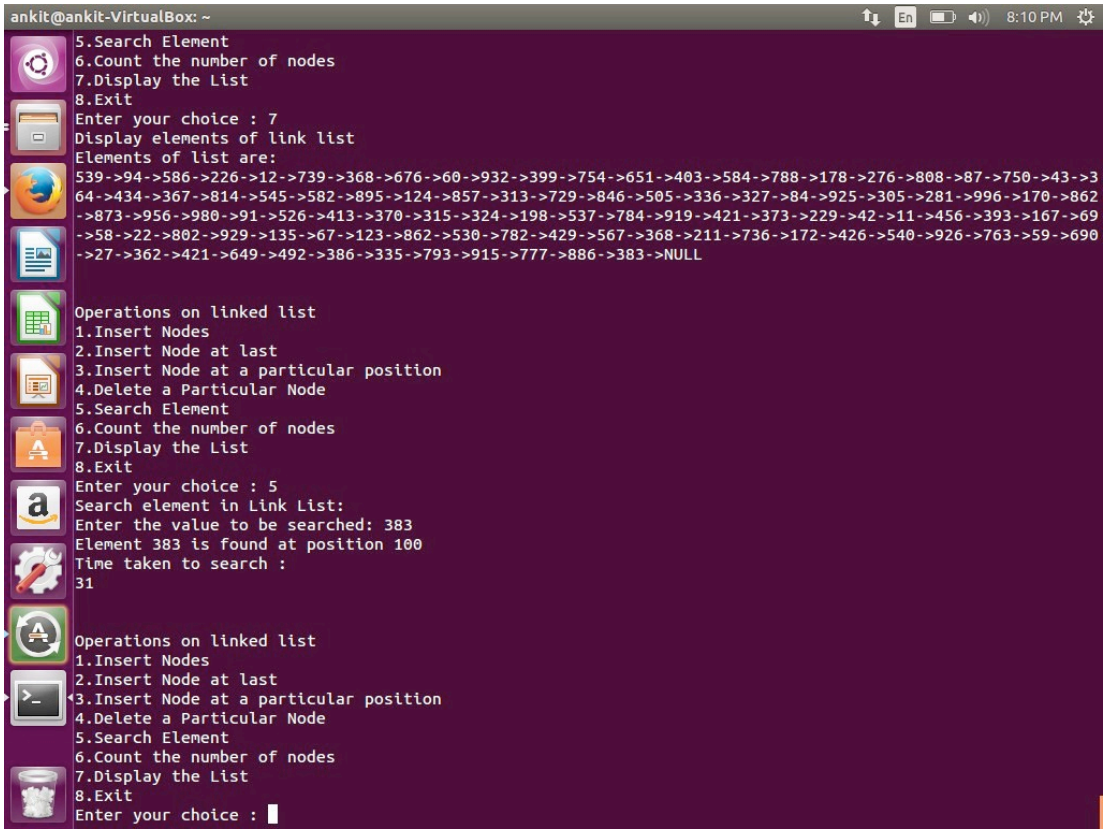
```
        {
            flag = true;
            cout<<"Element "<<value<<" is found at position "<<pos<<endl;
        }
        s = s->next;
    }
    if (!flag)
        cout<<"Element "<<value<<" not found in the list"<<endl;
}
 /* counting the number of elements */
void single_llist::count()
{
    struct node *temp;
    int length = 0;
    temp = initialize;
    while(temp!=NULL)
    {
        length++;
        temp=temp->next;
    }
printf("Length of Linked List : %d\n",length);
}



/*Display Elements of a link list*/

void single_llist::display()

{

    struct node *temp;

    if (initialize == NULL)

    {

        cout<<"The List is Empty"<<endl;

        return;

    }

    temp = initialize;
```

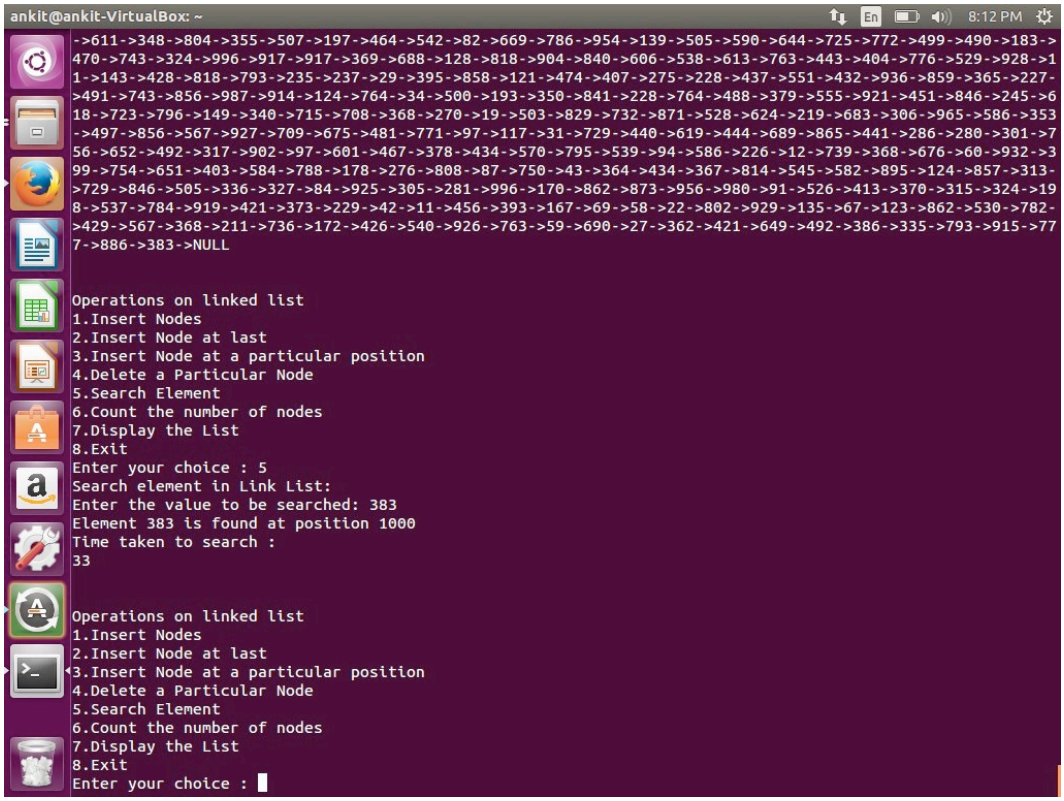
```
    cout<<"Elements of list are: "<<endl;

    while (temp != NULL)

    {

        cout<<temp->info<<"->";

        temp = temp->next;

    }

    cout<<"NULL"<<endl;

}

/*Main :contains menu */

main()

{

    int choice, nodes, element, position, i;
    single_llist sl;
    initialize = NULL;
    while (1)

    {

              cout<<endl<<"Operations on linked list"<<endl;
              cout<<"1.Insert Nodes"<<endl;
              cout<<"2.Insert Node at last"<<endl;
              cout<<"3.Insert Node at a particular position"<<endl;
              cout<<"4.Delete a Particular Node"<<endl;
              cout<<"5.Search Element"<<endl;
              cout<<"6.Count the number of nodes"<< endl;
              cout<<"7.Display the List"<<endl;
              cout<<"8.Exit "<<endl;
              cout<<"Enter your choice : ";

        cin>>choice;

        switch(choice)

        {

        case 1:

            cout<<"Inserting number of Nodes: "<<endl;
                      sl.insert_begin();
                      cout<<endl;

            break;
```

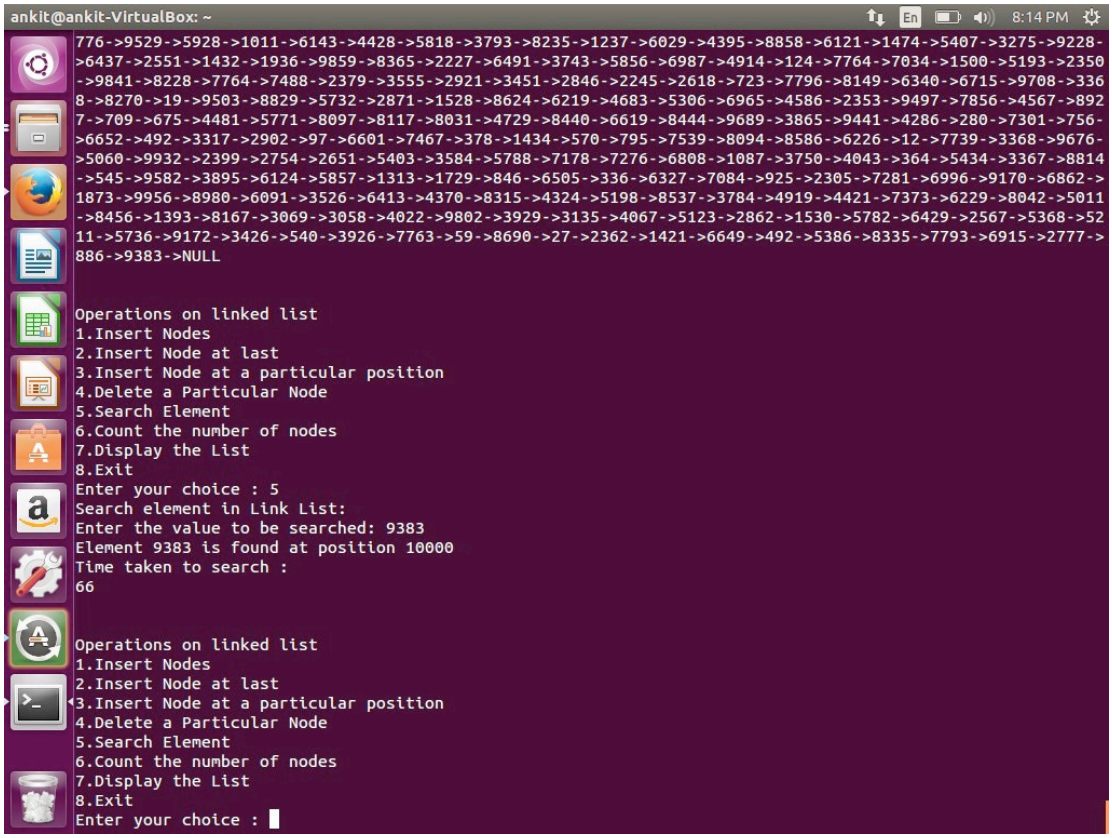
```
case 2:

    cout<<"Inserting Node at Last: "<<endl;
              sl.insert_last();
              cout<<endl;

    break;

case 3:

    cout<<"Inserting Node at a given position:"<<endl;
              sl.insert_pos();
              cout<<endl;

    break;


case 4:

    cout<<"Delete a particular node: "<<endl;
              sl.delete_pos();
              break;


case 5:

    cout<<"Search element in Link List: "<<endl;
      t1 = clock();
              sl.search();
      t2 = clock();
      cout<<"Time taken to search :\n"<<(t2-t1)<<endl;
              cout<<endl;

    break;

case 6:

    cout<<"Count the number of nodes: "<<endl;

    sl.count();

    cout<<endl;

    break;

case 7:

    cout<<"Display elements of link list"<<endl;
              sl.display();
              cout<<endl;

    break;

case 8:
```

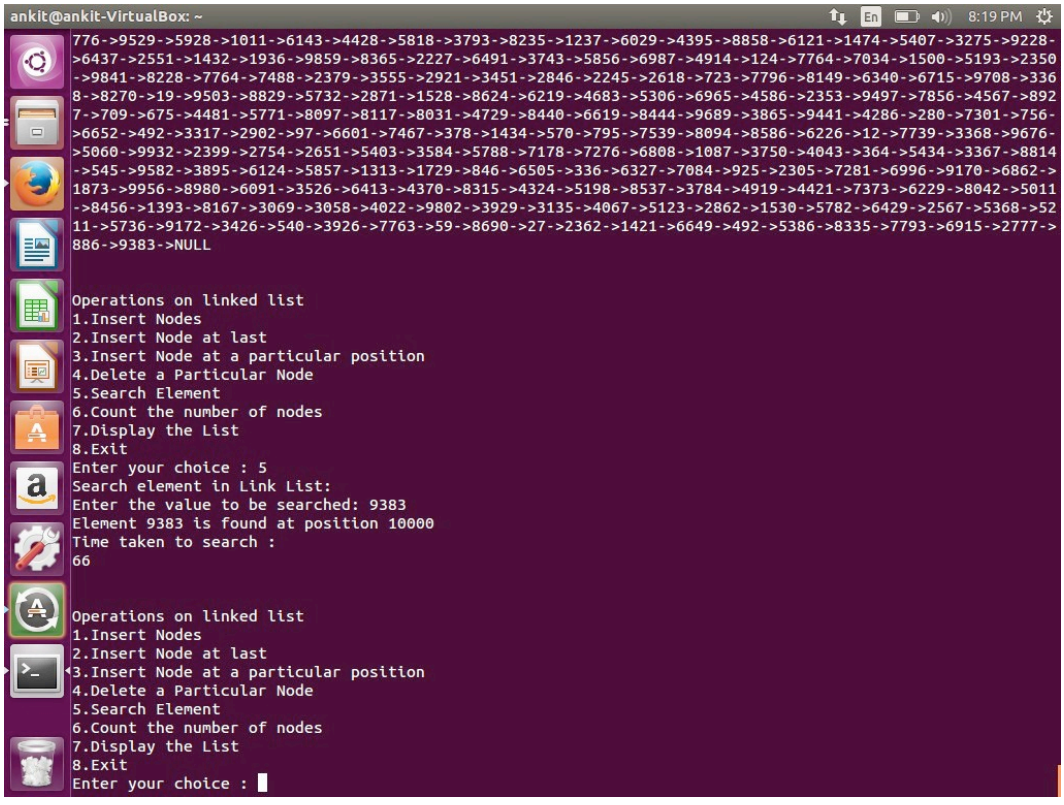
```
            cout<<"Exiting..."<<endl;
                      exit(1);

            break;


        default:

            cout<<"choice not in option"<<endl;

        }

    }

}
```

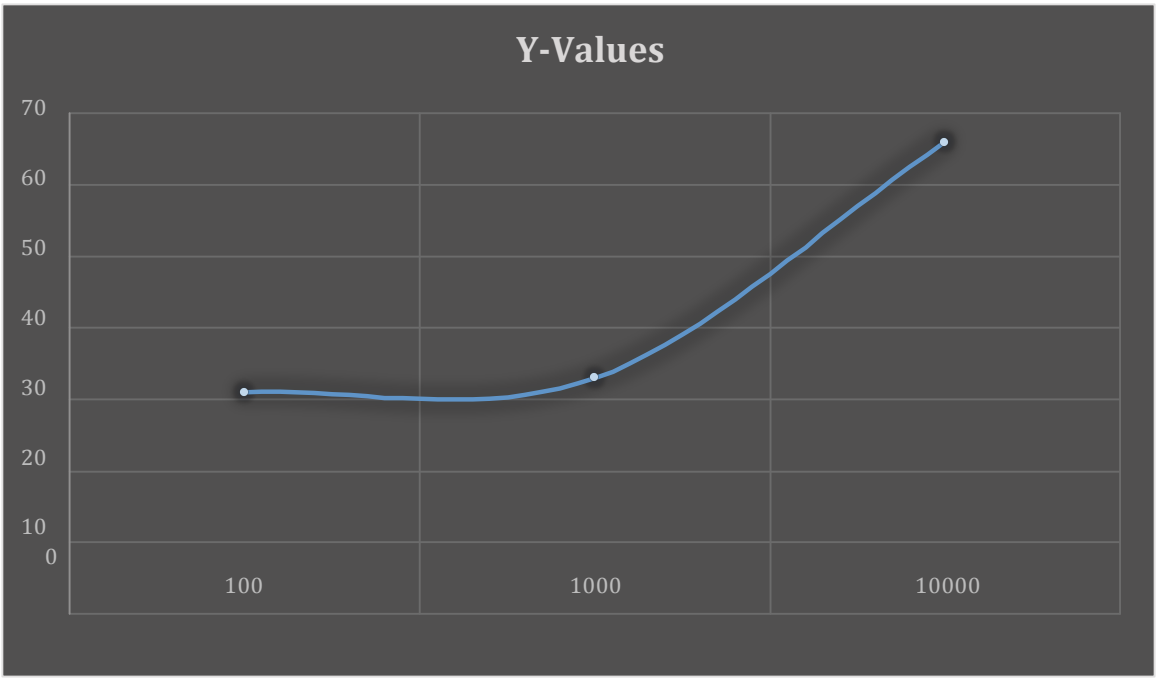
Timer Output

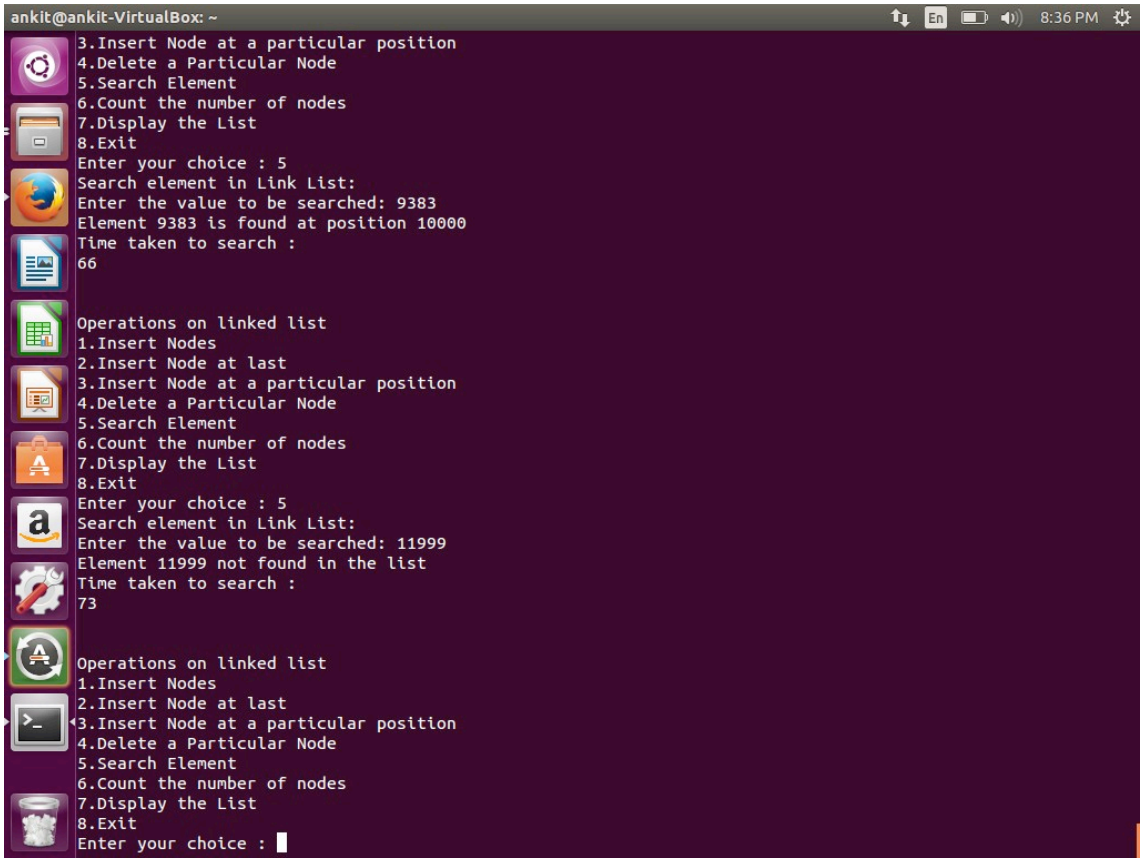



ankit@ankit-VirtualBox: ~
8:12 PM
->611->348->804->355->507->197->464->542->82->669->786->954->139->505->590->644->725->772->499->490->183->
470->743->324->996->917->917->369->688->128->818->904->840->606->538->613->763->443->404->776->529->928->1
1->143->428->818->793->235->237->29->395->858->121->474->407->275->228->437->551->432->936->859->365->227-
>491->743->856->987->914->124->764->34->500->193->350->841->228->764->488->379->555->921->451->846->245->6
18->723->796->149->340->715->708->368->270->19->503->829->732->871->528->624->219->683->306->965->586->353
->497->856->567->927->709->675->481->771->97->117->31->729->440->619->444->689->865->441->286->280->301->7
56->652->492->317->902->97->601->467->378->434->570->795->539->94->586->226->12->739->368->676->60->932->3
99->754->651->403->584->788->178->276->808->87->750->43->364->434->367->814->545->582->895->124->857->313-
>729->846->505->336->327->84->925->305->281->996->170->862->873->956->980->91->526->413->370->315->324->19
8->537->784->919->421->373->229->42->11->456->393->167->69->58->22->802->929->135->67->123->862->530->782-
>429->567->368->211->736->172->426->540->926->763->59->690->27->362->421->649->492->386->335->793->915->77
7->886->383->NULL
Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice : 5
Search element in Link List:
Enter the value to be searched: 383
Element 383 is found at position 1000
Time taken to search :
33
Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice :


ankit@ankit-VirtualBox: ~
8:14 PM
776->9529->5928->1011->6143->4428->5818->3793->8235->1237->6029->4395->8858->6121->1474->5407->3275->9228->6437->2551->1432->1936->9859->8365->2227->6491->3743->5856->6987->4914->124->7764->7034->1500->5193->2350->9841->8228->7764->7488->2379->3555->2921->3451->2846->2245->2618->723->7796->8149->6340->6715->9708->3368->8270->19->9503->8829->5732->2871->1528->8624->6219->4683->5306->6965->4586->2353->9497->7856->4567->8927->709->675->4481->5771->8097->8117->8031->4729->8440->6619->8444->9689->3865->9441->4286->280->7301->756->6652->492->3317->2902->97->6601->7467->378->1434->570->795->7539->8094->8586->6226->12->7739->3368->9676->5060->9932->2399->2754->2651->5403->3584->5788->7178->7276->6808->1087->3750->4043->364->5434->3367->8814->545->9582->3895->6124->5857->1313->1729->846->6505->336->6327->7084->925->2305->7281->6996->9170->6862->1873->9956->8980->6091->3526->6413->4370->8315->4324->5198->8537->3784->4919->4421->7373->6229->8042->5011->8456->1393->8167->3069->3058->4022->9802->3929->3135->4067->5123->2862->1530->5782->6429->2567->5368->5211->5736->9172->3426->540->3926->7763->59->8690->27->2362->1421->6649->492->5386->8335->7793->6915->2777->886->9383->NULL
Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice : 5
Search element in Link List:
Enter the value to be searched: 9383
Element 9383 is found at position 10000
Time taken to search :
66
Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice :


ankit@ankit-VirtualBox: ~
8:19 PM
776->9529->5928->1011->6143->4428->5818->3793->8235->1237->6029->4395->8858->6121->1474->5407->3275->9228->6437->2551->1432->1936->9859->8365->2227->6491->3743->5856->6987->4914->124->7764->7034->1500->5193->2350->9841->8228->7764->7488->2379->3555->2921->3451->2846->2245->2618->723->7796->8149->6340->6715->9708->3368->8270->19->9503->8829->5732->2871->1528->8624->6219->4683->5306->6965->4586->2353->9497->7856->4567->8927->709->675->4481->5771->8097->8117->8031->4729->8440->6619->8444->9689->3865->9441->4286->280->7301->756->6652->492->3317->2902->97->6601->7467->378->1434->570->795->7539->8094->8586->6226->12->7739->3368->9676->5060->9932->2399->2754->2651->5403->3584->5788->7178->7276->6808->1087->3750->4043->364->5434->3367->8814->545->9582->3895->6124->5857->1313->1729->846->6505->336->6327->7084->925->2305->7281->6996->9170->6862->1873->9956->8980->6091->3526->6413->4370->8315->4324->5198->8537->3784->4919->4421->7373->6229->8042->5011->8456->1393->8167->3069->3058->4022->9802->3929->3135->4067->5123->2862->1530->5782->6429->2567->5368->5211->5736->9172->3426->540->3926->7763->59->8690->27->2362->1421->6649->492->5386->8335->7793->6915->2777->886->9383->NULL
Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice : 5
Search element in Link List:
Enter the value to be searched: 9383
Element 9383 is found at position 10000
Time taken to search :
66
Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice :

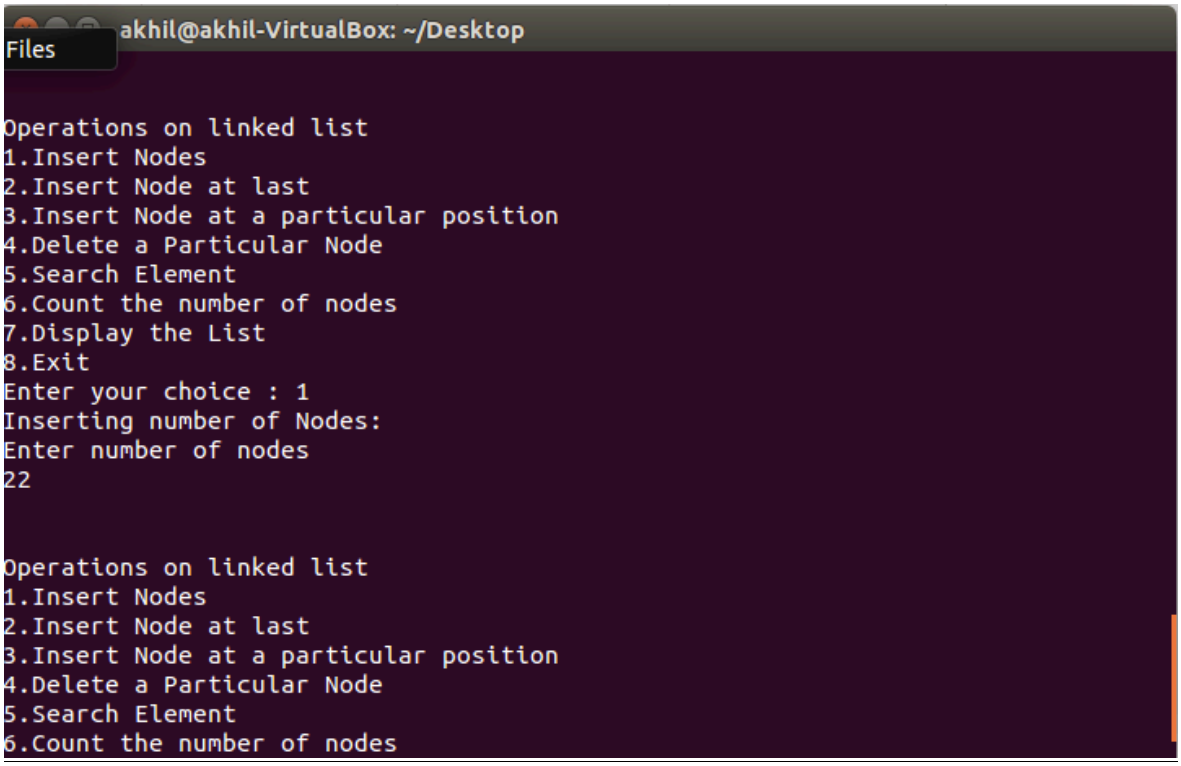




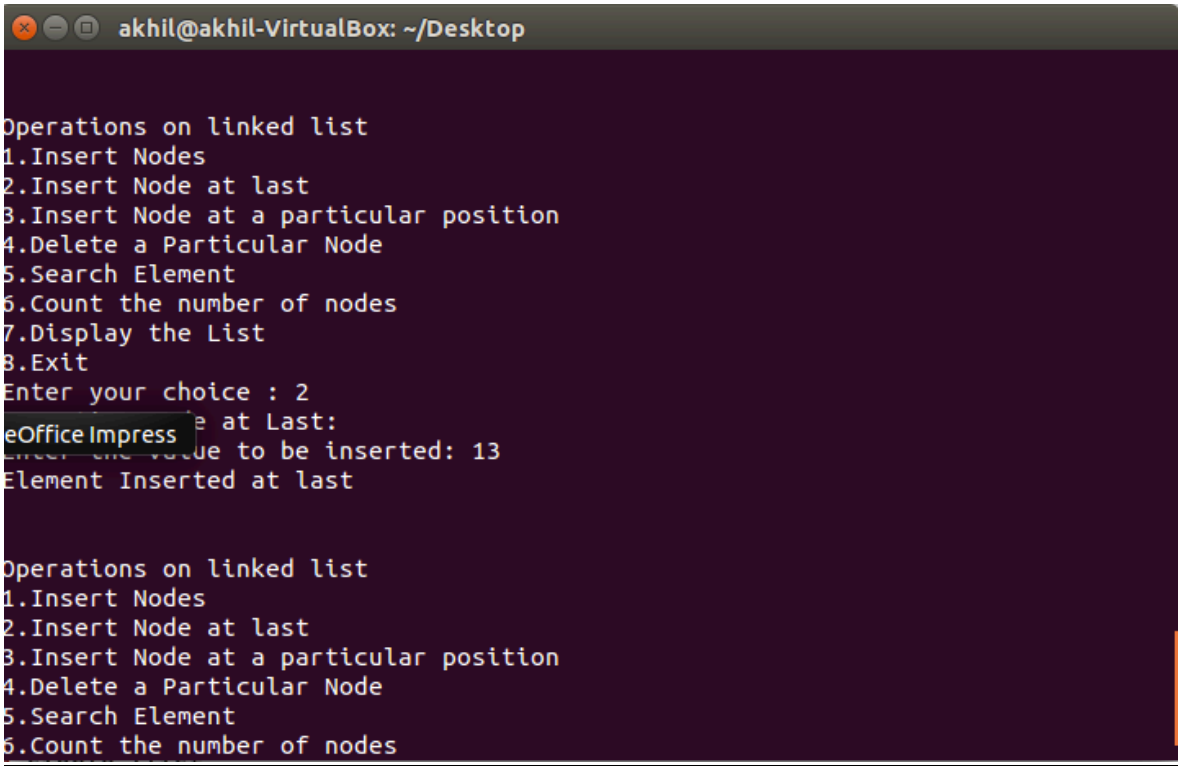
The above graph is for a non exisiting value !

Program output :

1.inserting input.



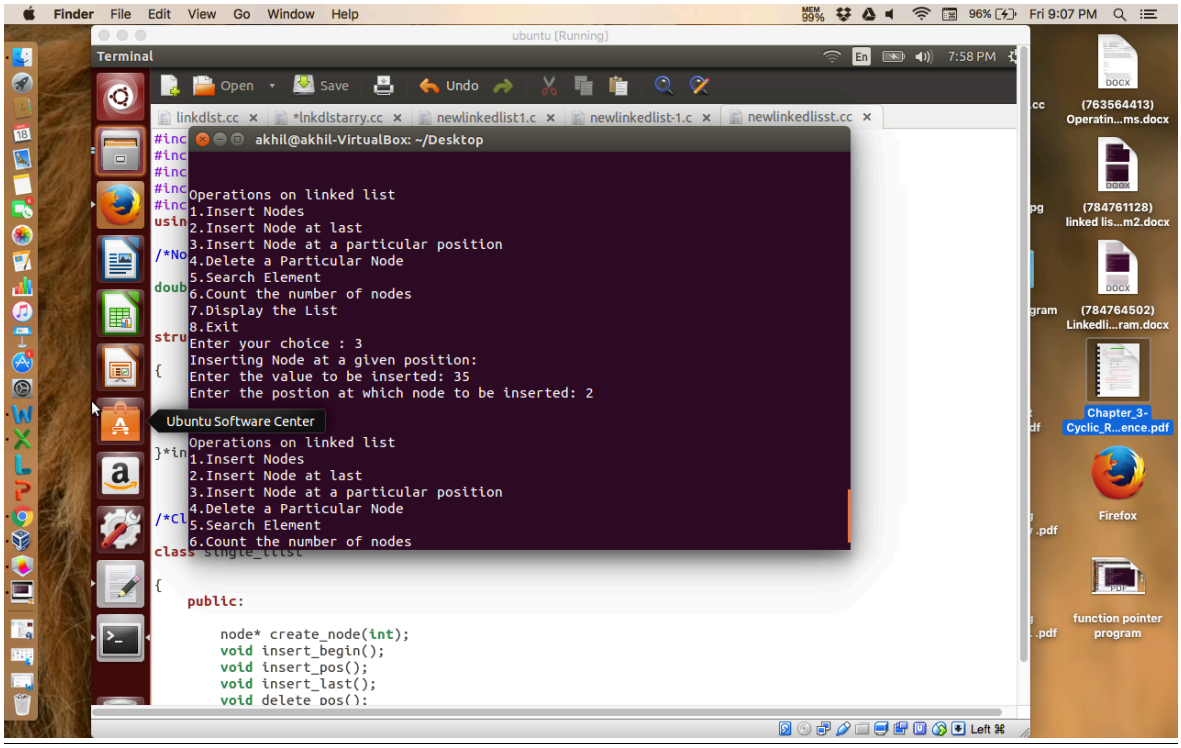
2.Inserting node at last.

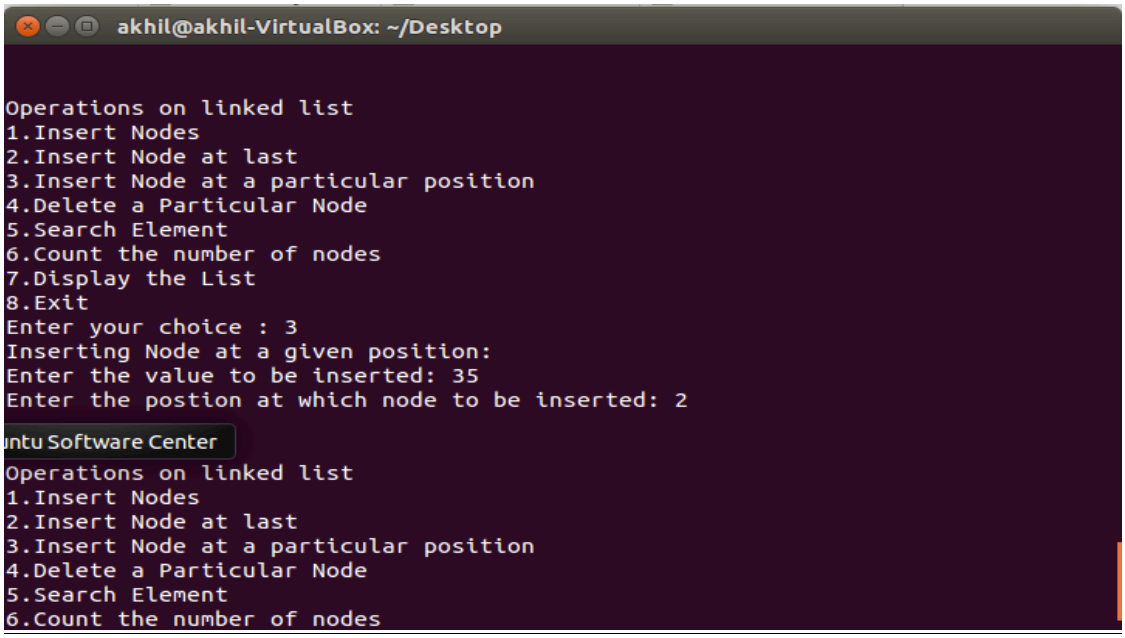


3. Display the list

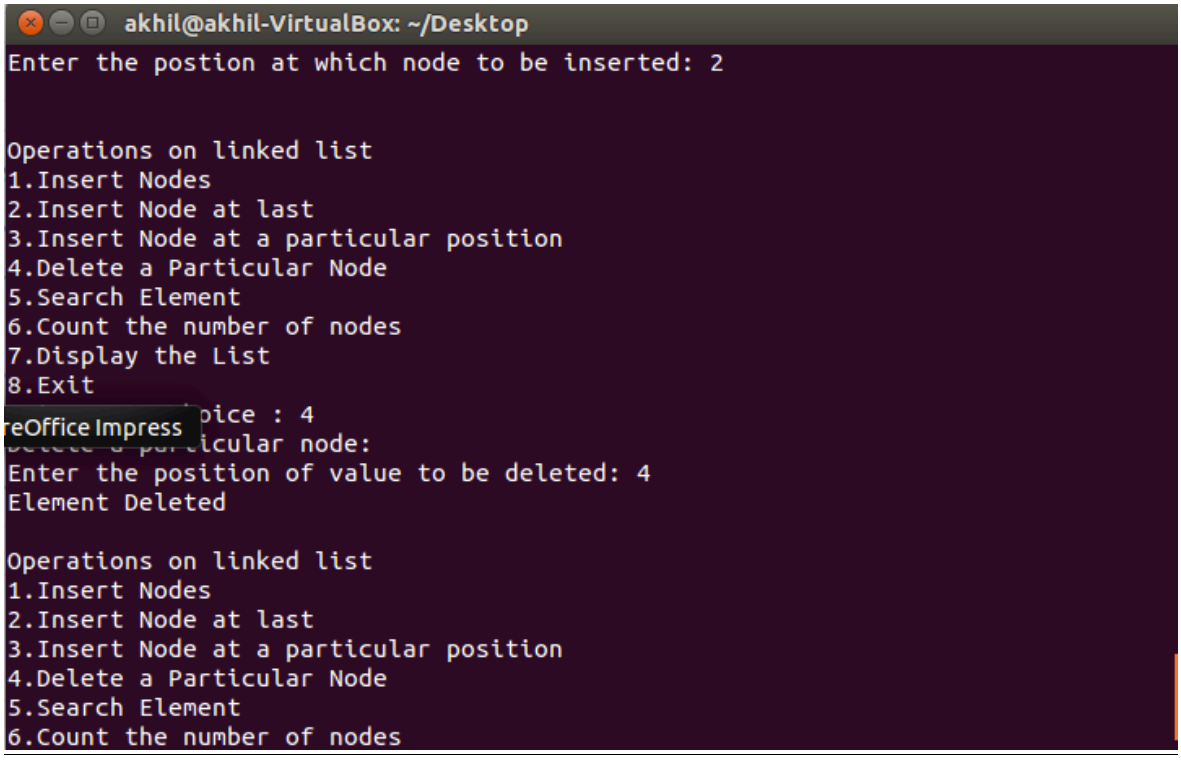
```
akhil@akhil-VirtualBox: ~/Desktop

Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
7.Display the List
8.Exit
Enter your choice : 7
Display elements of link list
eOffice Impress
736->9172->3426->540->3926->7763->59->8690->27->2362->1421->6649->4
92->5386->8335->7793->6915->2777->886->9383->13->NULL


Operations on linked list
1.Insert Nodes
2.Insert Node at last
3.Insert Node at a particular position
4.Delete a Particular Node
5.Search Element
6.Count the number of nodes
```

4.Insert node at a particular location.

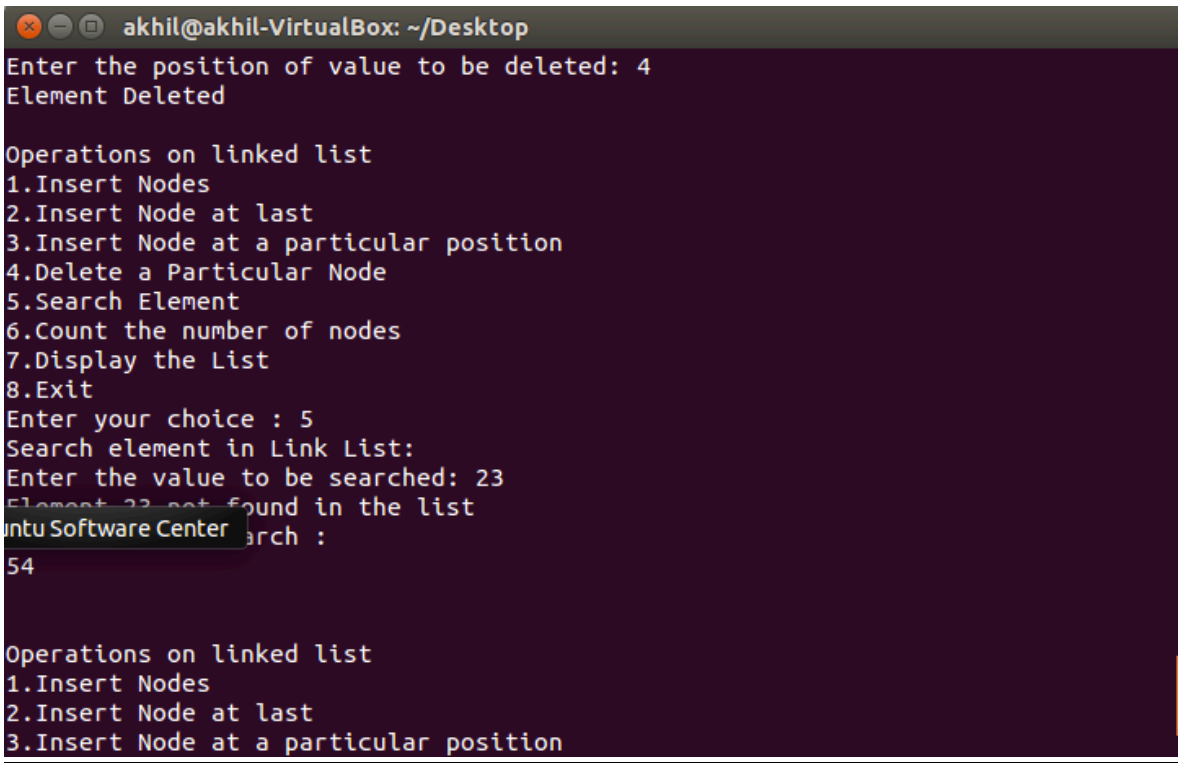




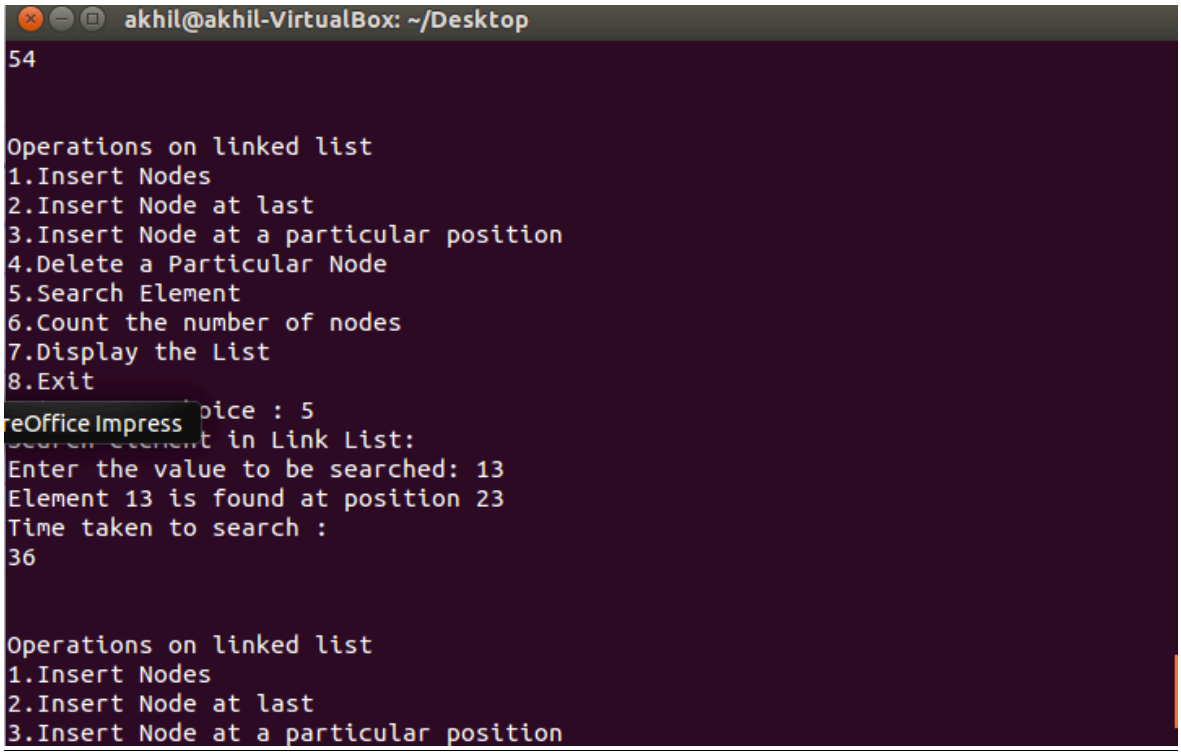
the above output is for inserting at a particular location.



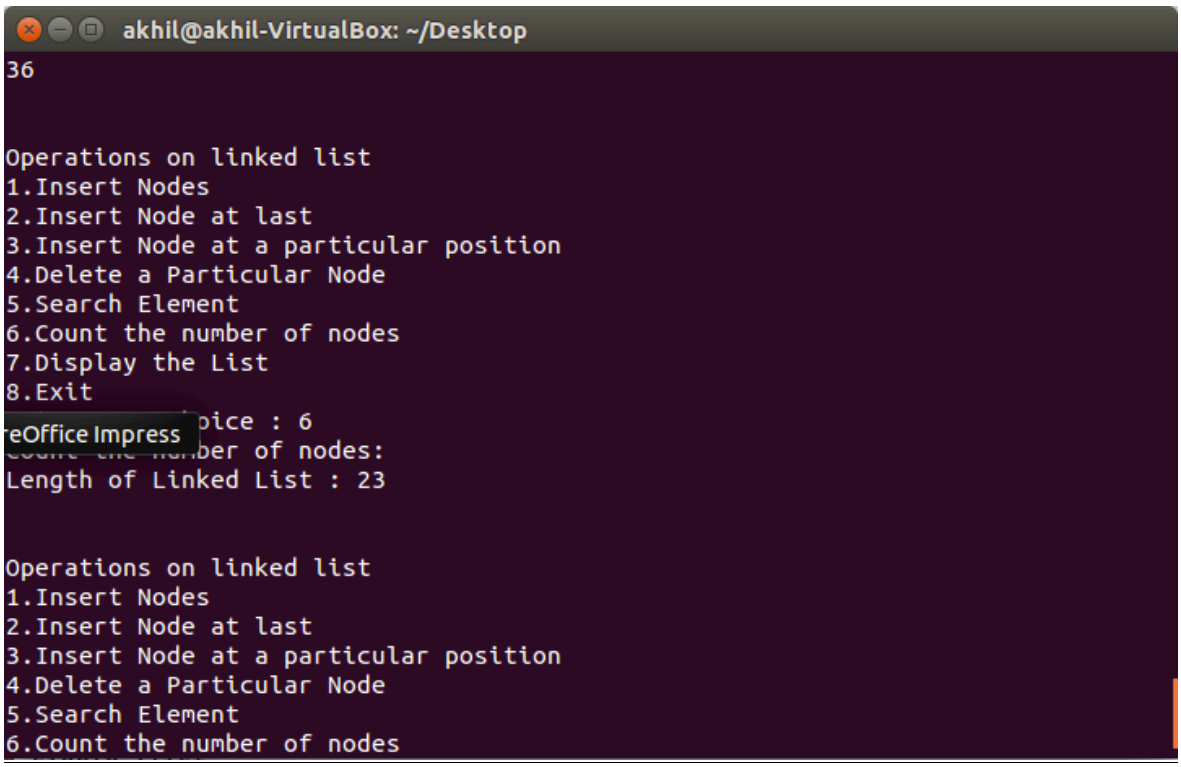
the above output is for deleting a particular node .



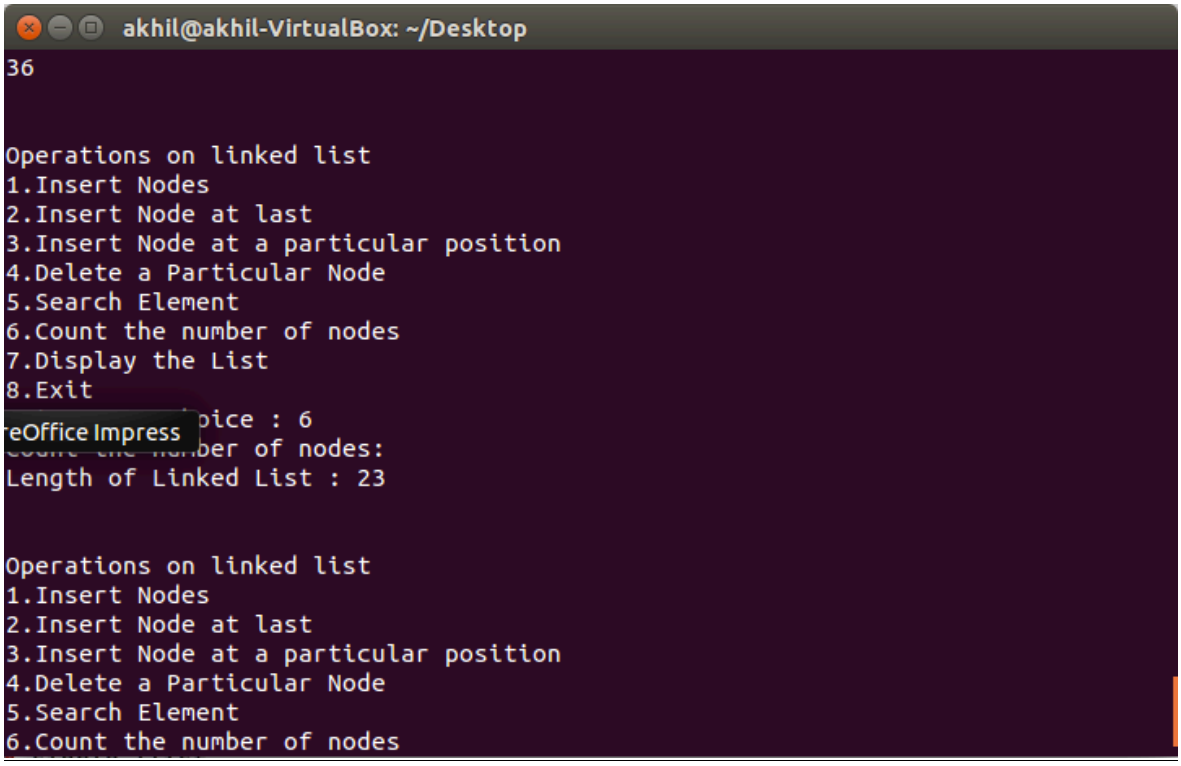
the above output is for searching a particular node.

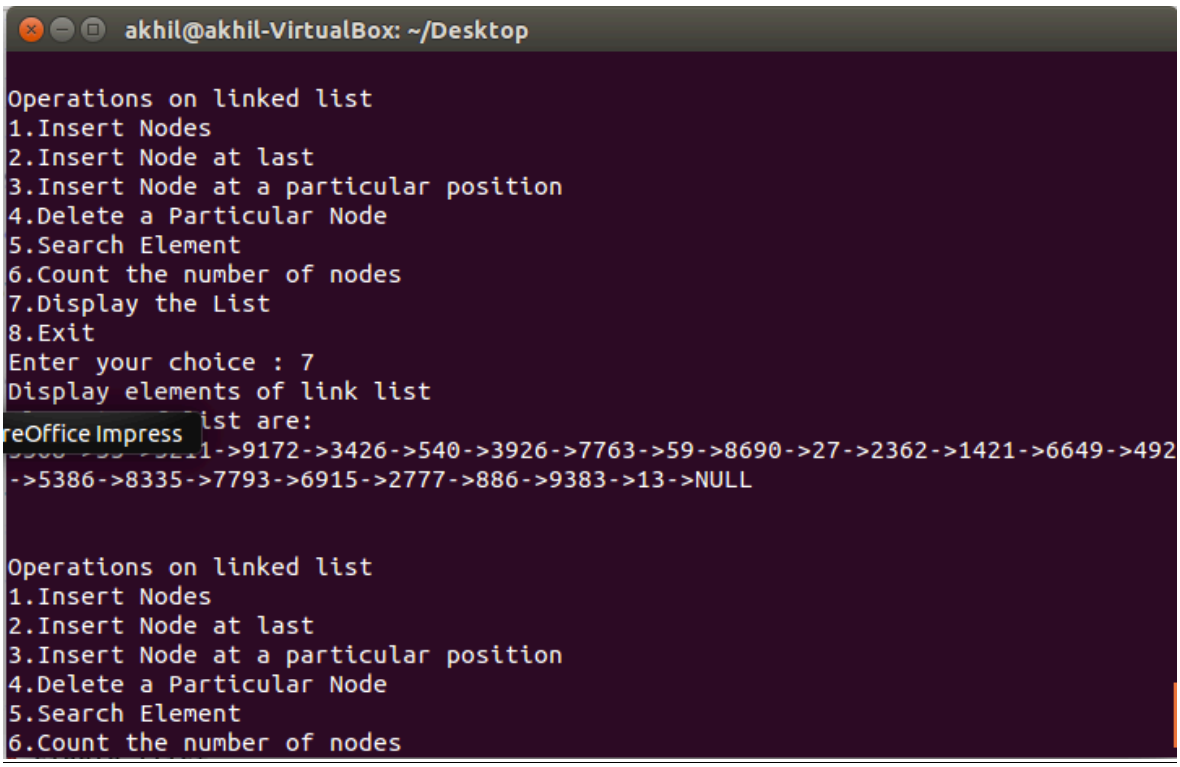


the above output is when an element is not found !

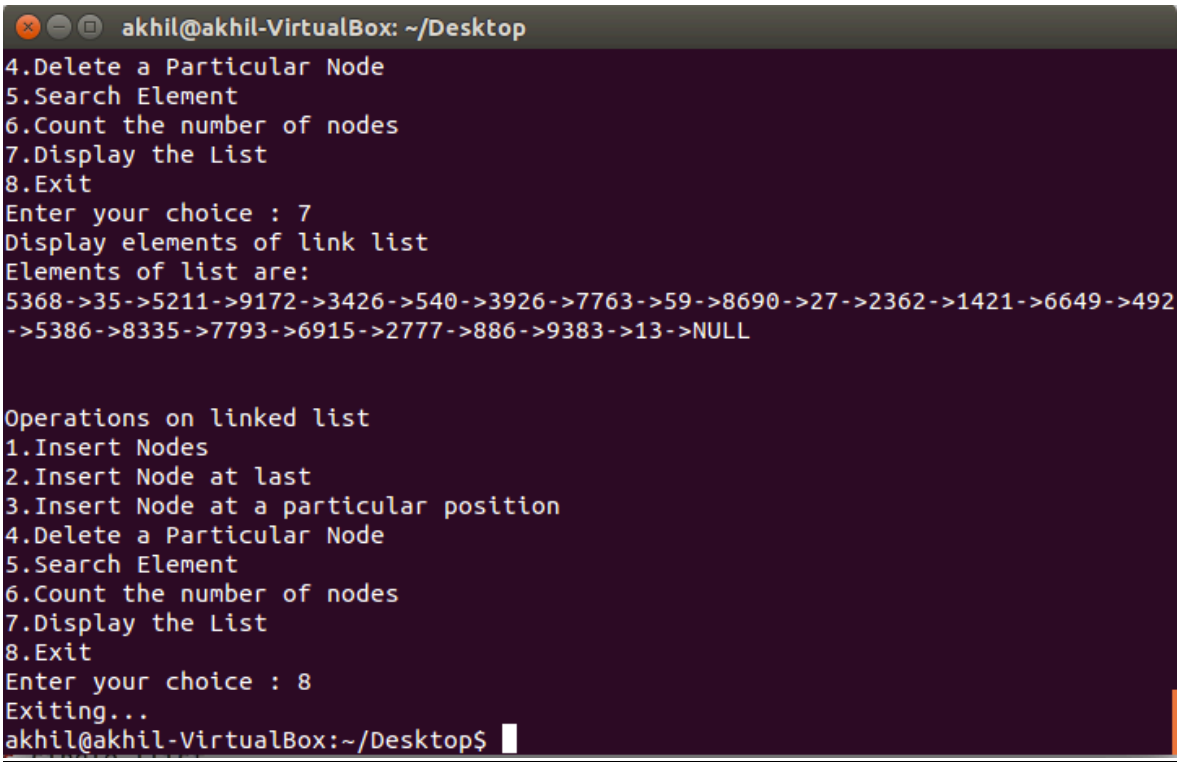


the above output is to count the number of nodes.





output : display the list
!!



output : exit ....

Group: Spring 20

Student name: Akhil Pal
Student ID: 010694840
Student name : Sanman Pradhan
Student ID: 010825295