**Practical 6**

**Aim: Implement following operations of singly linked list.**

**(a) Insert a node at front**

**(b) Insert a node at end**

**(c) Insert a node after given node information**

**(d) Delete a node at front**

**(e) Delete a node at last**

**(f) Delete a node after given node information**

**(g) Display link list**

**Algorithm:**

1. **Insert a node at front:**

n<= take data part of new node

head = insert\_front(n); // where n is data part of node

1. nod1 = create\_node(n)
2. if head == NULL

then return nod1

1. else

nod1 -> link = head

return nod1

1. **Insert a node at end:**

insert\_end(n); // where n is data part of node

1. nod1 = create\_node(n)
2. if head == NULL

then head = nod1

1. else

set p = head

while((p->link)! = NULL)

p = p -> link;

end while

p -> link = nod1;

1. **Insert a node after given node information:**

insert\_after();

1. if head == NULL

then print (First you have to create a node because link list is empty now)

end if

1. temp = 0;
2. ele <= take information of node after you want to insert new node
3. n <= take data of new node
4. nod1 = create\_node(n)
5. set p = head
6. while ((p -> data)! = ele && (p -> link! = NULL))

p = p -> link

if((p->data) == ele)

then temp = 1;

end if

end while

1. if temp == 1 || (head->data) == ele

then set q = p -> link

p -> link = nod1

nod1 -> link = q

temp = 1

end if

1. if temp == 0

then printf("Entered data is not found in Link List.\n")

end if

1. **Delete a node at front:**

delete\_front();

1. if head == NULL

then print (Link list is empty)

end if

1. else if ((head -> link == NULL))

head = NULL

end else if

1. else head = head -> link
2. **Delete a node at end:**

delete\_last();

1. if head == NULL

then print (Link list is empty)

end if

1. else if ((head -> link == NULL))

head = NULL

end else if

1. set p = head, q
2. while((p->link)! = NULL)

q = p;

p = p -> link;

end while

1. q -> link = NULL
2. **Delete a node after given node information:**

delete\_after();

1. temp = 0;
2. if head == NULL

then print (Link List is empty)

end if

1. ele <= take information of of node
2. set p = head
3. while ((p -> data)! = ele && (p -> link! = NULL))

p = p -> link

if((p->data) == ele)

then temp = 1;

end if

end while

1. if p->link == NULL

then printf ((Data is not fond) OR (after given data there is no node))

end if

1. if temp == 1 || (head->data) == ele

then p -> link = p -> link -> link

end if

1. **Display link list:**

disply();

1. set dis = head
2. if head == NULL

then print (Link list is empty)

end if

1. while (dis! = NULL)

printf ("[%d - %u] ", dis->data, dis->link)

dis = dis -> link;

end while

**Tracing of Link List:**

1. **Insert a node at front:**

|  |  |
| --- | --- |
| 23 | \0 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 21 | \0 |

Suppose link list is: and you want to insert a node in the front of given link list. And address of new node is 098 and head=099.

head = insert\_front(21)

1. nod1 = create\_node(21) = 098
2. if head == NULL ………False
3. else

098 -> link = 099

return 099

so, head = 099

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Now, link-list is:

1. **Insert a node at end:**

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Suppose link list is:

|  |  |
| --- | --- |
| 24 | \0 |

And head = 098. And you want to add new node is:

insert\_end(24);

1. nod1 = create\_node(24) = 200
2. if head == NULL ………... False
3. else

set p = head = 098

while ((098->link)! = NULL) ……………. True

p = 098 -> link = 099;

while ((099->link)! = NULL) ……………. True

p = 099 -> link = 100;

while((100->link)! = NULL) …………... False

end while

100 -> link = 200;

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

|  |  |
| --- | --- |
| 24 | \0 |

Now link list is:

1. **Insert a node after given node information:**

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Suppose link list is:

|  |  |
| --- | --- |
| 24 | \0 |

And head = 098. And you want to add new node is:

After the node which data part is 22. So, ele = 22

1. if head == NULL ……………. False
2. temp = 0;
3. ele <= 22 // take information of node after you want to insert.
4. n <= 24 //take data of new node
5. nod1 = create\_node(24)
6. set p = head = 098
7. while ((098 -> data)! = 22 && (098 -> link! = NULL)) …………True

p = 098 -> link = 099

if ((099->data) == 22) …………… True

temp = 1

while ((099 -> data)! = 22 && (099 -> link! = NULL)) …………False

end while

1. if temp == 1 || (head->data) == ele …………… True

then set q = 099 -> link = 100

099 -> link = 200

200 -> link = 100

temp = 1

end if

1. if temp == 0 ……………. False

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 200 |

|  |  |
| --- | --- |
| 24 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Now link list is:

1. **Delete a node at front:**

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Suppose link list is:

And head = 098. And you want to delete first node.

1. if head == NULL ………………...False
2. else if ((098 -> link == NULL)) …………. False
3. else head = 098 -> link = 099

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

So, head = 099 and New link list is:

1. **Delete a node at end:**

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Suppose link list is:

And head = 098. And you want to delete last node.

1. if 098 == NULL ………………...False
2. else if ((098 -> 099 == NULL)) …………. False
3. set p = head = 098, q
4. while((098->link)! = NULL) ………... True

q = 098;

p = 098 -> link = 99

while((099->link)! = NULL) ………... True

q = 099;

p = 099 -> link = 100

while((100->link)! = NULL) ………... False

end while

1. 099 -> link = NULL

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | \0 |

So, Now link list is:

1. **Delete a node after given node information:**

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | 200 |

|  |  |
| --- | --- |
| 24 | \0 |

Suppose link list is:

And head = 098. And you want to delete a node after node which data part is 22.

1. temp = 0;
2. if head == NULL ………………...False
3. ele <= 22 //take information of of node
4. set p = head =098
5. while ((098 -> data)! = 22 && (098 -> link! = NULL)) …………True

p = 098 -> link = 099

if((099->data) == 22) …………True

then temp = 1;

end if

while ((099 -> data)! = 22 && (098 -> link! = NULL)) …………False

end while

1. if 099->link == NULL ……………...False
2. if temp == 1 || (head->data) == ele

then 099 -> link = 099 -> link -> link = 200

end if

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 200 |

|  |  |
| --- | --- |
| 23 | \0 |

So, Now link list is:

1. **Display link list:**

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 100 |

|  |  |
| --- | --- |
| 23 | \0 |

Suppose link list is:

And head = 098. And you want to display this link list.

1. set dis = head = 098
2. if head == NULL …………... False
3. while (098! = NULL) ……………. True

printf ("[21 - 099] ", 098->data, 098->link)

dis = 098 -> link = 099

while (099! = NULL) ……………. True

printf ("[22 - 100] ", 099->data, 099->link)

dis = 099 -> link = 100

while (100! = NULL) ……………. True

printf ("[23 - NULL] ", 100->data, 100->link)

dis = 100 -> link = NULL

while (NULL! = NULL) ……………. False

end while

|  |  |
| --- | --- |
| 21 | 099 |

|  |  |
| --- | --- |
| 22 | 200 |

|  |  |
| --- | --- |
| 23 | \0 |

So, it prints

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*link;

}\*head=NULL;

struct node\* create\_node(int n)

{

struct node\* newNode;

newNode = (struct node\*)malloc(sizeof(struct node));

newNode -> data = n;

newNode -> link = NULL;

return newNode;

}

struct node\* insert\_front(int n)

{

struct node\* nod1;

nod1 = create\_node(n);

if(head == NULL)

return nod1;

else

{

nod1 -> link = head;

return nod1;

}

}

struct node\* insert\_end(int n)

{

struct node\* nod1;

nod1 = create\_node(n);

if(head == NULL)

head = nod1;

else

{

struct node\* p = head;

while((p->link)!= NULL)

{

p = p -> link;

}

p -> link = nod1;

}

printf("Node is inserted.\n");

}

struct node\* insert\_after()

{

if(head == NULL)

{

printf("First You have to Create a node because link list is empty now.");

}

else

{

int ele,n,temp=0;

printf("Enter the data after you want to display your node: ");

scanf("%d", &ele);

printf("Enter the element you want to insert: ");

scanf("%d",&n);

struct node\* nod1;

nod1 = create\_node(n);

struct node\* p = head;

struct node\* q;

while(((p -> data) != ele) && (p -> link != NULL))

{

p = p -> link;

if((p->data) == ele)

temp = 1;

}

if(temp == 1 || (head->data) == ele)

{

q = p -> link;

p -> link = nod1;

nod1 -> link = q;

printf("Node is inserted.\n");

temp = 1;

}

if(temp == 0)

printf("Entered data is not found in Link List.\n");

}

}

void delete\_front()

{

if(head == NULL)

printf("Link list is empty");

else if((head -> link == NULL))

{

head = NULL;

printf("Node is deleted.\n");

}

else

{

head = head -> link;

printf("Node is deleted.\n");

}

}

void delete\_last()

{

if(head == NULL)

printf("Link list is empty");

else if((head -> link == NULL))

{

head = NULL;

printf("Node is deleted.\n");

}

else

{

struct node\* p = head;

struct node\* q;

while((p->link) != NULL)

{

q = p;

p = p -> link;

}

q -> link = NULL;

printf("Node is deleted.\n");

}

}

struct node\* delete\_after()

{

int temp=0;

if(head == NULL)

{

printf("Link list empty.");

}

else

{

int ele,n;

printf("Enter that data after you want to delete your node: ");

scanf("%d", &ele);

struct node\* p = head;

while(((p -> data) != ele) && (p -> link )!= NULL)

{

p = p -> link;

if((p->data) == ele)

temp = 1;

}

if(p->link == NULL)

printf("(Data is not fond ) OR (After given data there is no node).\n");

if(temp == 1 || head->data == ele)

{

p -> link = p -> link -> link;

printf("Node is deleted.\n");

}

}

}

void disply()

{

struct node\* dis = head;

if(head == NULL)

printf("Link list is empty");

while(dis!=NULL)

{

printf("[%d - %u] ",dis->data,dis->link);

dis = dis -> link;

}

printf("\n");

}

int main()

{

int n,w;

printf("\n---------------------Menu----------------------");

printf("\n 1. Insert a node at front");

printf("\n 2. Insert a node at end");

printf("\n 3. Insert a node after given node information");

printf("\n 4. Delete a node at front");

printf("\n 5. Delete a node at last");

printf("\n 6. Delete a node after given data");

printf("\n 7. Display Link-List");

printf("\n 8. Exit");

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

do{

printf("\nEnter the number to select option of given menu: ");

scanf("%d",&w);

switch(w)

{

case 1:

printf("Enter the element: ");

scanf("%d",&n);

head = insert\_front(n);

printf("Node is inserted.\n");

break;

case 2:

printf("Enter the element: ");

scanf("%d",&n);

insert\_end(n);

break;

case 3:

insert\_after();

break;

case 4:

delete\_front();

break;

case 5:

delete\_last();

break;

case 6:

delete\_after();

break;

case 7:

disply();

break;

case 8:

exit(0);

break;

default:

printf("Enter the correct choice");

break;

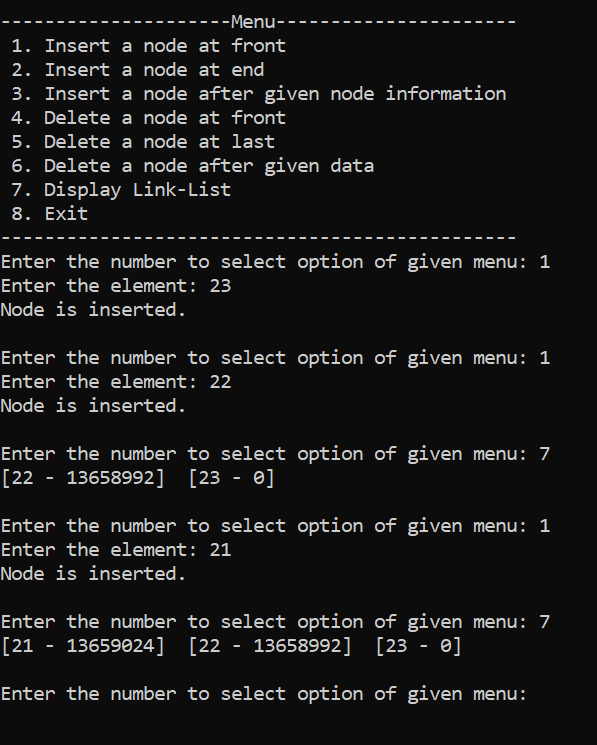
}

}while(w!=8);

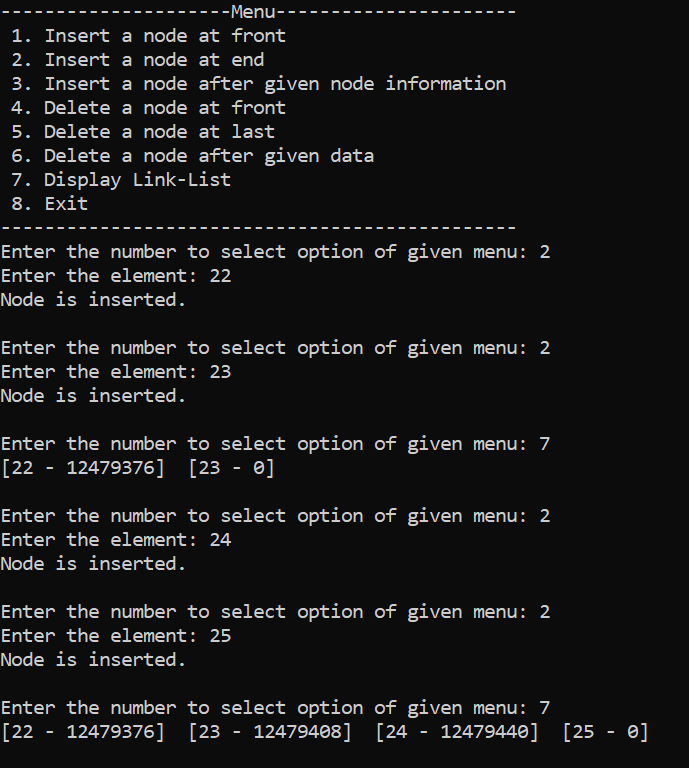
}

**Output:**

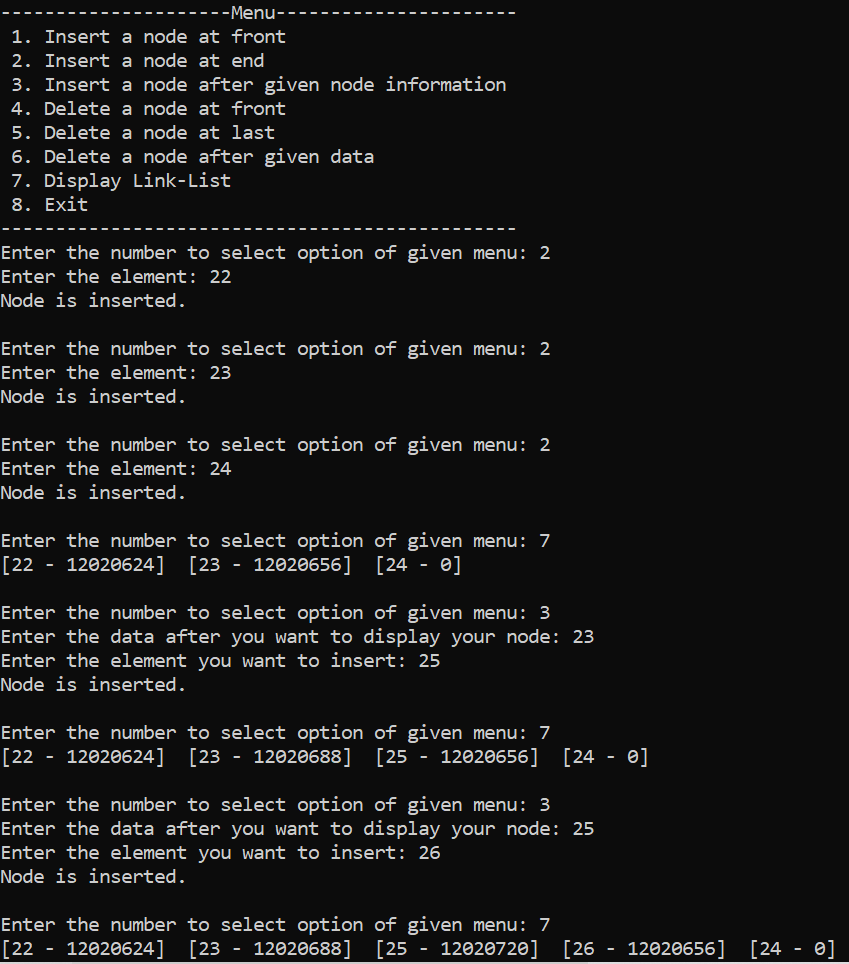
1. Insert a node at front:



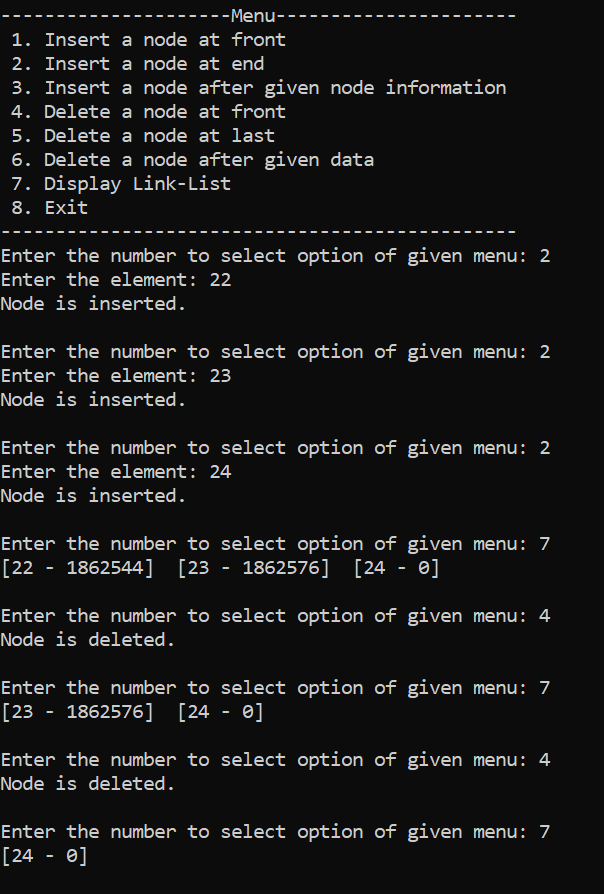
1. Insert a node at end:



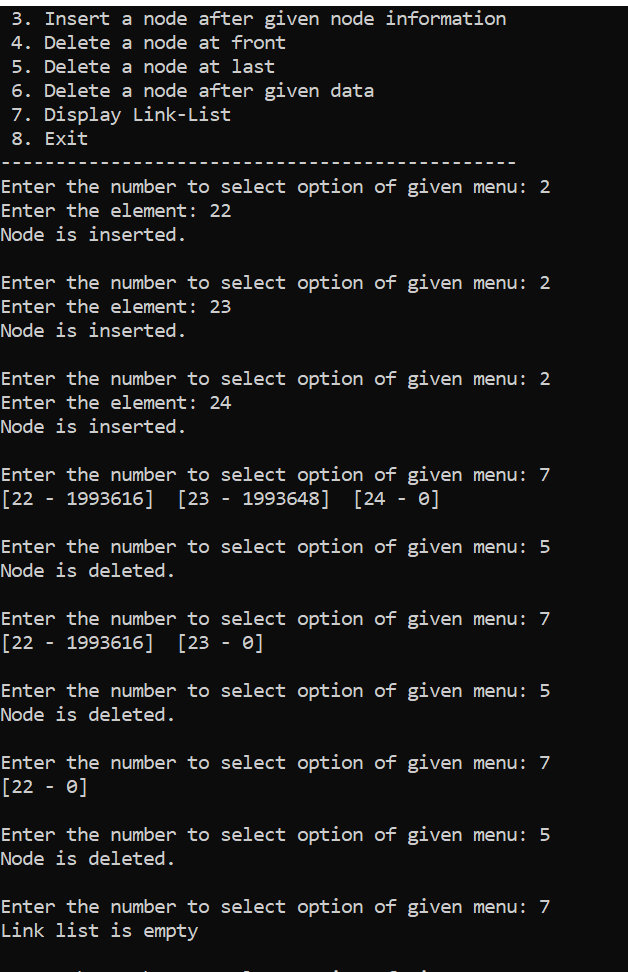
1. Insert a node after given node information:



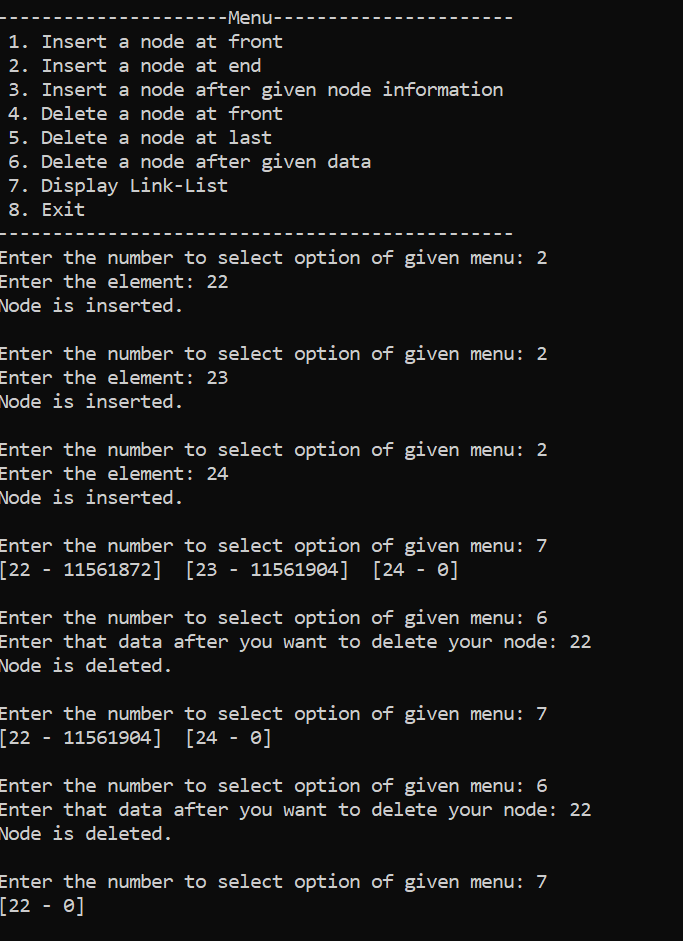
1. Delete a node at front:



1. Delete a node at end:



1. Delete a node after given node information:



1. Display link list:

