```
#include<iostream>
#include<stdlib.h>
using namespace std;
- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -


- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
template <class T>
class LinkedList
{

// -------------------------- definition of node structure -------------------------- //
        class Node
        {
                        friend class LinkedList;

                        T data;
                        Node *link;
                public:
                        Node ( T val )          // constructor of class node
                        {
                                data = val;
                                link = NULL;
                        }
        }
        *head, *tail;
        int size;

// ------------------- ========================== ------------------- //

        public:

                LinkedList()

                {
                        head =NULL;
                        tail= NULL;
                        size=0;
                }

                int isEmpty();
                void makeEmpty();
                void addHead(T item);
                T removeHead();
                void addTail(T item);
                T removeTail();
                void insert (int p, T item);
                void display();
                T remove(int p);
                int find (T item);
                T findKth(int k);
};
- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
int LinkedList <T> :: isEmpty()
{
        if (size==0) return 1;
        else return 0;
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
void LinkedList<T> :: makeEmpty()
{
        size=0;
        head=tail=NULL;
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
void LinkedList<T> :: addHead(T item)
{
        Node *newnode;

        newnode = new Node(item);
        newnode->link = head;
        head = newnode;

        if ( tail == NULL ) tail = head;
        ++size;

        cout << ".....Inserted..... : " << item;
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
T LinkedList <T> :: removeHead()
{
        Node *temp = head;

        T item = head->data;
        head = head->link;

        if ( head == NULL ) tail = NULL;
        --size;

        delete temp;
        return item;
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
void LinkedList <T> :: addTail(T item)
{
        if ( isEmpty() )
                                addHead(item);
        else
        {
                Node *newnode;

                newnode = new Node(item);
                tail->link = newnode;
                tail = newnode;

                ++size;

                cout << ".....Inserted..... : " << item;
        }
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
T LinkedList <T> :: removeTail( )
{
        if ( head==tail)
                        removeHead( );
        else
        {
                Node *temp = head;

                while ( temp->link != tail )
                {
                        temp = temp->link;
                }
                temp->link = NULL ;

                T item = tail->data;
                delete tail;
                tail = temp;
                --size;
                return item;
      }
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -



- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
template <class T>
void LinkedList <T> :: insert(int n, T item)
{
        if ( n == 1 )
                        addHead(item);

        else if ( n == size + 1 )

                        addTail(item);

        else if ( n > size+1 )

                cout << ".....Linked List is Smaller than SIZE.....!!";
        else
        {
                Node *newnode = new Node(item);
                Node *temp = head;

                for ( int k = 1; k < n-1; ++k )
                {
                        temp = temp->link;
                }

                newnode->link = temp->link;
                temp->link=newnode;
                ++size;

                cout << ".....Inserted..... : " << item;
        }
}
- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -



- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
template <class T>
void LinkedList <T> :: display()
{
        if( isEmpty() )
                        cout<<".....Linked List is EMPTY.....!! MSG from display()";
        else
        {
                Node *temp=head;

                while(temp!=NULL)
                {
                        cout << " -> " << temp->data;
                        temp = temp->link;
                }
        }
}
- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
T LinkedList <T> :: remove(int n)
{
        If ( isEmpty() ) cout << ".....linked List is EMPTY.....!!";

        else if ( n>size ) cout << ".....Linked List contain only " << size << " Elements.....";

                else  if ( n == 1 )  removeHead();

                else if ( n == size ) removeTail( );

                else
        {
                Node *temp = head;
                for (int k = 1; k < n-1; k++ )
                {
                        temp = temp->link;
                }

                Node * temp2 = temp->link;

                T item = temp2->data;
                temp->link = temp2->link;

                delete temp2;
                --size;
                return item;
      }
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

```
template <class T>
int LinkedList <T> :: find(T key)
{
        Node *temp=head;
        while(temp!=NULL)
        {
                if ( temp->data == key ) return 1;
                temp = temp->link;
        }
        return -1;
}
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
```
template <class T>
T LinkedList <T> :: findKth(int k)
{
	if ( k>size ) return -1;
	else
	{
		Node *temp=head;

		for ( int i=1; i<k; ++i )
		{
			temp = temp->link;
		}
	return temp->data;
	}
}
```
- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -
```
main()
{
	int ch, k;
	LinkedList <int> list;
	int item, pos;
	do
	{
		cout << "\n..........Linked List ADT..........";
		cout << "\n1...AddHead  \n2...AddTail \n3...RemoveHead  \n4...RemoveTail";
		cout << "\n5...Insert At Middle \n6...Delete From Middle\n7...Display \n8...Make Empty";
		cout << "\n9...Find\n10..Find Kth Element \n11...Exit \n.....Enter Choice..... ? ";
		cin >> ch;
		switch (ch)
		{
			case 1:
				cout <<".....Enter the Element..... ? ";
				cin >> item;
				list.addHead(item);
				break;
			case 2:
				cout <<".....Enter the Element..... ? ";
				cin >> item;
				list.addTail(item);
				break;
			case 3:
				item=list.removeHead();
				cout<<".....Deleted..... : " << item;
				break;
```

```
                case 4:
                        item=list.removeTail();
                        cout<<".....Deleted..... : "<<item;
                        break;
                case 5:
                        cout <<".....Enter the Element & Position..... ? ";
                        cin >> item >> pos;
                        list.insert(item, pos);
                        break;
                case 6:
                        cout <<".....Enter the Position..... ? ";
                        cin >> pos;

                        item=list.remove(pos);
                        cout<<".....Deleted..... : " << item;
                        break;
                case 7:
                        list.display();
                        break;
                case 8:
                        list.makeEmpty();
                        break;
                case 9:
                        cout << ".....Enter the KEY..... ? ";
                        cin >> k;
                        pos=list.find(k);
                        if ( pos > 0 )
                                cout<<".....KEY Found in the Linked List...!!";
                        else
                                cout<<".....KEY NOT Present in the Linked List...!!";
                        break;
                case 10:
                        cout <<".....Enter Position to Search, K..... ? ";
                        cin >> k;
                        pos=list.findKth(k);
                        if ( pos>0 )
                                        cout<<".....The Element at Kth Position is..... : " << pos;
                        else
                                        cout<<".....The Linked List is Smaller than given K.....!!";
                        break;
                case 11:
                                exit(0);
            }       // END OF switch (ch)
        } while (ch!=11);       // END OF do-while
        return (0);
}       // END OF main()
```

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -

- ---------- ---------- ---------- ---------- ------------------- ---------- ---------- ---------- ---------- ---------- -