LinkedList

1

```
class Node{
    //node contain two below things
    int data;
    Node* next;

public:
    //so here data = d and by default pointer asign as null in this
constructor
    Node(int d):data(d),next(NULL){}
};

class List{
    Node * head;
    Node * tail;
public:
    List():head(NULL),tail(NULL){}
    void push_front(int data){
        if(head==NULL){
            Node * n = new Node(data);
            head = tail = n;
        }
        else{
            Node * n = new Node(data);
            n->next = head;
            head = n;
        }
    }
}
```

1

```
//in the array and vector the memory is going to be continuous memory
//while in the linked list it is on demand memory
#include <bits/stdc++.h>
using namespace std;
class Node{
public:
    int data;
    Node *next;
    Node(int data){
        this->data=data;
        next = NULL;
    }
};
int main(){
```

```
    //statically
    Node n1(1);
    Node n2(2);
    n1.next = &n2;
    cout << n1.data <<" " <<n2.data<< endl;
    Node * head = &n1; //it is containing the address of the first node
    cout << head->data;  //it is derefrenece goto head address and print the
data
    //dynamically
    Node *n3 = new Node(3);
    Node *n4 = new Node(4);
    n3->next = n4;
    return 0;

}
```

2

```
#include <bits/stdc++.h>
using namespace std;
class Node{
public:
    int data;
    Node *next;
    Node(int data){
        this->data=data;
        next = NULL;
    }
};
void print(Node * head){
    while(head!=NULL){
        cout << head->data <<"->";
        head = head->next;
    }
    cout<<"NULL"<<endl;
}
int main(){
    //statically
    Node n1(1);
    Node * head = &n1;
    Node n2(2);
    Node n3(3);
    Node n4(4);
    Node n5(5);
    n1.next = &n2;
    n2.next = &n3;
    n3.next = &n4;
    n4.next = &n5;
```

```
    print(head);
    return 0;
}
```

3

```
#include <bits/stdc++.h>
using namespace std;
class Node{
public:
    int data;
    Node *next;
    Node(int data){
        this->data=data;
        next = NULL;
    }
};
void print(Node * head){
    Node * temp = head;
    while(temp!=NULL){
        cout << temp->data <<"->";
        temp = temp->next;
    }
    cout<<"NULL"<<endl;
}
int length(Node * head){
    int count = 0;
    Node * temp = head;
    while(temp!=NULL){
        count++;
        temp = temp->next;
    }
    return count;
}
void returnIthElement(Node * head,int i){
    Node * temp = head;
    for(int j=0;j<i;j++){
        if(temp==NULL){
            return;
        }
        temp = temp->next;
    }
    cout << temp->data << endl;
}
// Node * insertAtIthElement(Node * head,int i,int insert){
//  Node * temp = head;
//  for(int j=0;j<i;j++){
//      if(temp==NULL){
```

```
//          return;
//      }
//      temp = temp->next;
//  }
//  Node * n = new Node(insert);
//  n->next = temp->next;
//  temp->next = n;
//  return head;
// }
Node * takeInput(){
    //no constraint in the size
    int data;
    cin >> data;
    Node * head = NULL; //ll is empty
    Node * tail = NULL; //LL is empty
    while(data!=-1){
        //creating LL
        //here we can not create LL staticaly because it is valid till the
itration
        //so we have to use the dyanmicaly
        Node * n = new Node(data);
        //1st node or not
        if(head==NULL){
            head = n;
            tail = n;
        }else{
            tail->next = n;
            tail = n;
        }
        cin >> data;
    }
    return head;
}
Node * takeInputReverse(){
    //no constraint in the size
    int data;
    cin >> data;
    Node * head = NULL; //ll is empty
    Node * tail = NULL; //LL is empty
    while(data!=-1){
        //creating LL
        //here we can not create LL staticaly because it is valid till the
itration
        //so we have to use the dyanmicaly
        Node * n = new Node(data);
        //1st node or not
        if(head==NULL){
            head = n;
```

```
            tail = n;
        }else{
            n->next = head;
            head = n;
        }
        cin >> data;
    }
    return head;
}
int main(){
    Node * head = takeInput();
    Node * head2 = takeInputReverse();
    print(head);
    print(head2);
    // insertAtIthElement(head,2,4);
    // print(head);
    // returnIthElement(head,3);
    // int x= length(head);
    // cout << x << endl;
    return 0;

}
```

4

```
#include <bits/stdc++.h>
using namespace std;
class Node{
public:
    int data;
    Node *next;
    Node(int data){
        this->data=data;
        next = NULL;
    }
};
void print(Node * head){
    Node * temp = head;
    while(temp!=NULL){
        cout << temp->data <<"->";
        temp = temp->next;
    }
    cout<<"NULL"<<endl;
}
int length(Node * head){
    int count = 0;
    Node * temp = head;
    while(temp!=NULL){
```

```
        count++;
        temp = temp->next;
    }
    return count;
}
void returnIthElement(Node * head,int i){
    Node * temp = head;
    for(int j=0;j<i;j++){
        if(temp==NULL){
            return;
        }
        temp = temp->next;
    }
    cout << temp->data << endl;
}
Node * takeInput(){
    //no constraint in the size
    int data;
    cin >> data;
    Node * head = NULL; //ll is empty
    Node * tail = NULL; //LL is empty
    while(data!=-1){
        //creating LL
        //here we can not create LL staticaly because it is valid till the
itration
        //so we have to use the dyanmicaly
        Node * n = new Node(data);
        //1st node or not
        if(head==NULL){
            head = n;
            tail = n;
        }else{
            tail->next = n;
            tail = n;
        }
        cin >> data;
    }
    return head;
}
void insertAtIthElement(Node * head,int i,int insert){
    if(head==NULL){
            return;
        }
    for(int j=0;j<i;j++){
        if(head==NULL){
            return;
        }
        head = head->next;
```

```
    }
    if(i>=0){
    Node * n = new Node(insert);
    n->next = head->next;
    head->next = n;
}
}
Node * deleteIthElement(Node* head,int i){

    if(i<0){
        return head;
    }
    if(i==0 && head){
        Node * newHead = head->next;
        head->next=NULL;
        delete head;
        return newHead;
        //return head->next; //here we are returning the second node add and
1st node is going to delete
    }
    Node * curr = head;
    int count = 1;
    while(count<=i-1 && curr!=NULL){
        curr = curr->next;
        count++;
    }
    if(curr && curr->next){
        Node * temp = curr->next;
        curr->next = curr->next->next;
        temp->next = NULL;
        delete temp;
        return head;
    }
    return head;
}
int main(){
    Node * head = takeInput();
    print(head);
    insertAtIthElement(head,3,4);
    print(head);
    deleteIthElement(head,3);
    print(head);
    return 0;
}
```

5

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
class Node{
public:
    int data;
    Node * next;
    Node(int data){
        this->data = data;
        next = NULL;
    }
};
int lengthRec(Node * head){
    if(head==NULL) return 0;
    int smallAns = lengthRec(head->next);
    return 1 + smallAns;
}
Node * takeInput(){
    int data;
    cin >> data;
    Node * head = NULL;
    Node * tail = NULL;
    while(data!=-1){
        Node * n = new Node(data);
        if(head==NULL){
            head = n;
            tail = n;
        }else{
            tail->next = n;
            tail = n;
        }
        cin >> data;
    }
    return head;
}
bool isPresent(Node * head, int data){
    Node * curr = head;
    while(curr!=NULL){
        if(curr->data == data){
            return true;
        }
        curr = curr->next;
    }
    return false;
}
int main(){
    Node * head = takeInput();
    int ak = lengthRec(head);
    cout << ak <<endl;
    cout <<isPresent(head,4) << endl;
```

```
    return 0;

}
```

6

```
//find the mid of the ll
//so here we are taking the two pointers
//one is fast and second one is slow
//slow pointer pass one and fast one passes the two pass
//when fast reaches the end at that time slow one will reach at the mid
#include <bits/stdc++.h>
using namespace std;
class Node{
public:
    int data;
    Node * next;
    Node(int data){
        this->data = data;
        next = NULL;
    }
};
Node * takeInput(){
    int data;
    cin >> data;
    Node * head = NULL;
    Node * tail = NULL;
    while(data!=-1){
        Node * n = new Node(data);
        if(head==NULL){
            head = n;
            tail = n;
        }else{
            tail->next = n;
            tail = n;
        }
        cin >> data;
    }
    return head;
}
Node * middleNode(Node * head){
    Node * slow = head;
    Node * fast = head->next;
    while(fast && fast->next){
        slow = slow->next;
        fast = fast->next->next;
    }
    if(fast){ //even
```

```
        return slow->next;
    }
    return slow; //odd
}
void print(Node * head){
    Node * temp = head;
    while(temp!=NULL){
        cout << temp->data <<"->";
        temp = temp->next;
    }
    cout<<"NULL"<<endl;
}
int main(){
    Node * head = takeInput();
    print(head);
    Node * output = middleNode(head);
    print(output);

}
```

7

```
//merge two sorted list
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* merge(ListNode* l1,ListNode* l2){
        if(l1==NULL){
            return l2;
        }
        if(l2==NULL){
            return l1;
        }
        if(l1->val < l2->val){
            l1->next = merge(l1->next,l2);
            return l1;
        }
        else{
            l2->next = merge(l1,l2->next);
```

```
            return l2;
        }
    }
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
      return merge(list1,list2);
    }
};
```

8

```
#include <bits/stdc++.h>
using namespace std;
class Node{
public:
    int data;
    Node * next;
    Node(int data){
        this->data = data;
        next = NULL;
    }
};
Node * takeInput(){
    int data;
    cin >> data;
    Node * head = NULL;
    Node * tail = NULL;
    while(data!=-1){
        Node * n = new Node(data);
        if(head==NULL){
            head = n;
            tail = n;
        }else{
            tail->next = n;
            tail = n;
        }
        cin >> data;
    }
    return head;
}
void print(Node * head){
    Node * temp = head;
    while(temp!=NULL){
        cout << temp->data <<"->";
        temp = temp->next;
    }
    cout<<"NULL"<<endl;
}
Node * reverseList(Node * head ){
```

```
}
int main(){

}
```

9

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int t;
    cin>>t;
    while(t--){
    int n;
    cin>>n;
    int h;
    cin>>h;
    int shot=0;
    for (int i = 1; i < n+1 ; ++i)
    {
        int x;
        cin>>x;
        if(x>h){
            shot++;
        }
    }
    cout<<shot<<endl;
    }
}
```

10

```
#include <bits/stdc++.h>

using namespace std;
int main(){
    int t;
    cin>>t;
    while(t--){
        string s1,s2;
        int length;
        cin>>length;
        cin>>s1;
        cin>>s2;
        int n=0;
        for (int i = 0; i < length; ++i)
        {
            if (s1[i]=='?')
```

```
            {
                s1[i]='j';
            }
            if (s2[i]=='?')
            {
                s2[i]='j';
            }
            if(s1[i]!=s2[i]){
                s2[i]=s1[i];
                n++;
            }
        }
        cout <<n <<endl;
    }
}
```