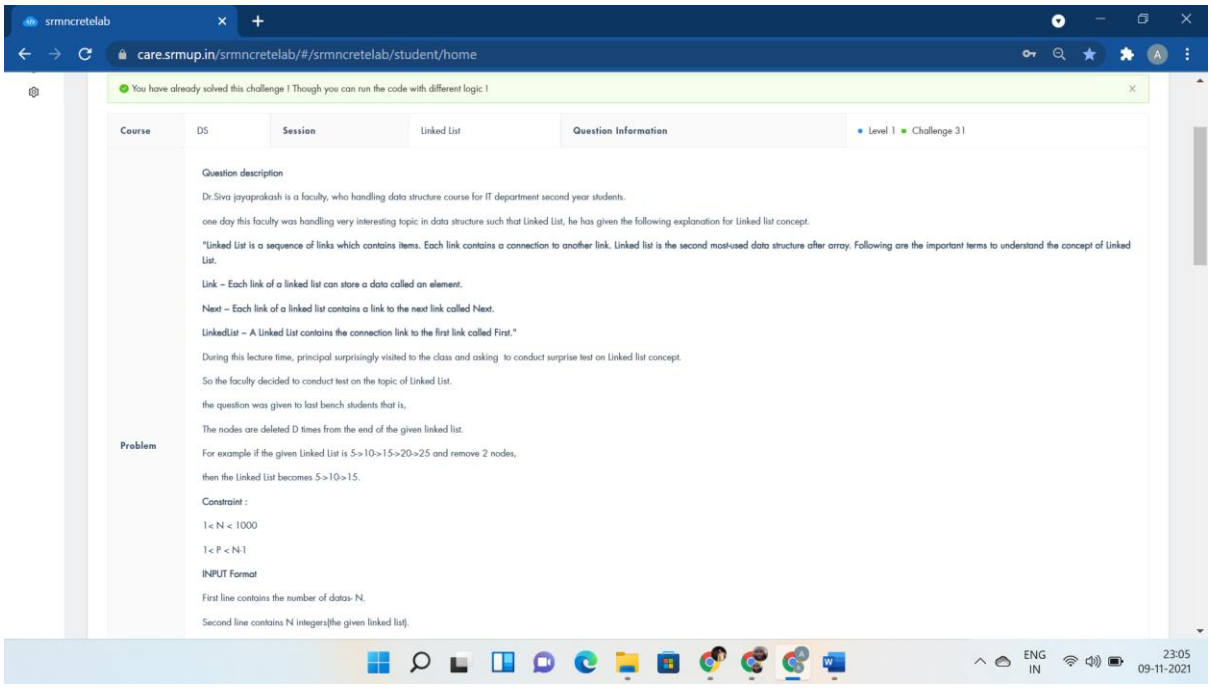
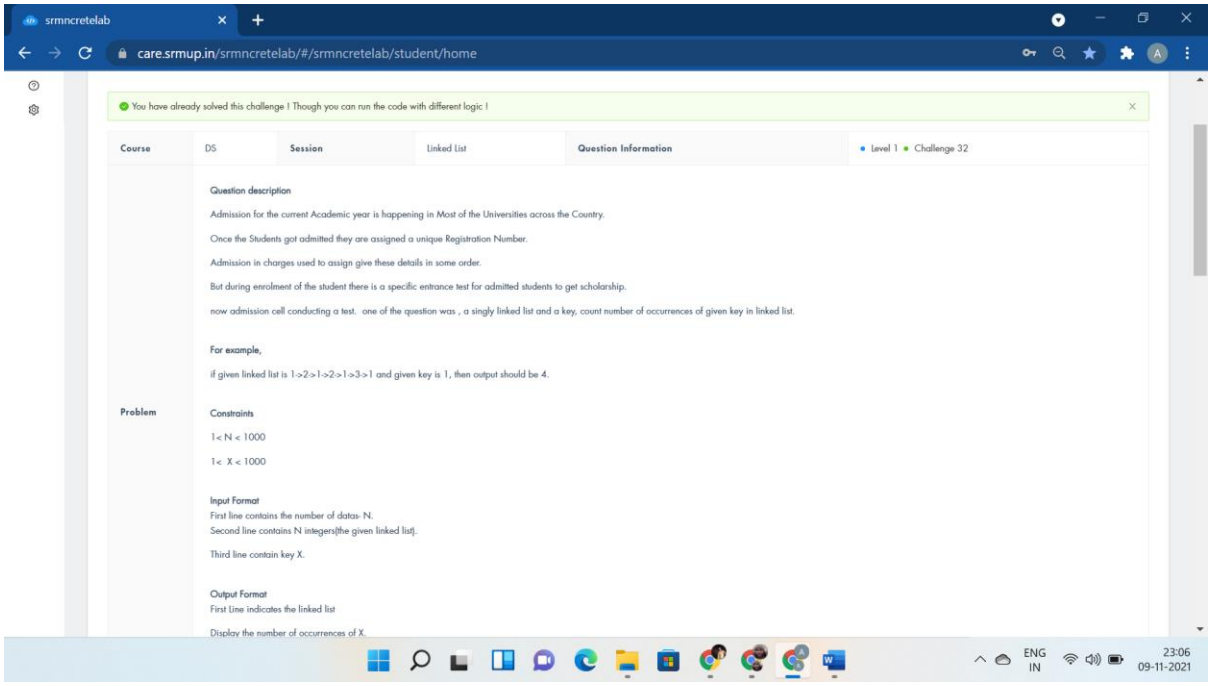
Linked List




```
#include <iostream>
using namespace std;
void tel(){
   return;}
struct node {
   int data;
   node *next;
}*head = NULL;
void create(){
   int n;
   cin >> n;
   struct node *p1 = new node;
   int m;
   cin >> m;
   p1->data = m;
   head = p1;
   int i;
   for (i = 0; i < n - 1; i++)  {
      int a;
```

```
        cin >> a;
        node *tt = new node;
        tt->data = a;
        p1->next = tt;
        p1=p1->next;
    }
    p1->next = NULL;
    int del;
    bool found = false;
    cin >> del;
    node *nn = head;
    while (nn != NULL) {
        nn = nn->next;
        node *dd = nn;
        int m = del;
        while (m-- > -1) {
            dd = dd->next;
            if (dd == NULL) {
                nn->next = NULL;
                found = true;
                break;
            }
        }
        if (found)
            break;
    }
    cout << "Linked List:";
    while (head != NULL){
        cout << "->" << head->data;
        head = head->next;
    }
}
int main(){
```

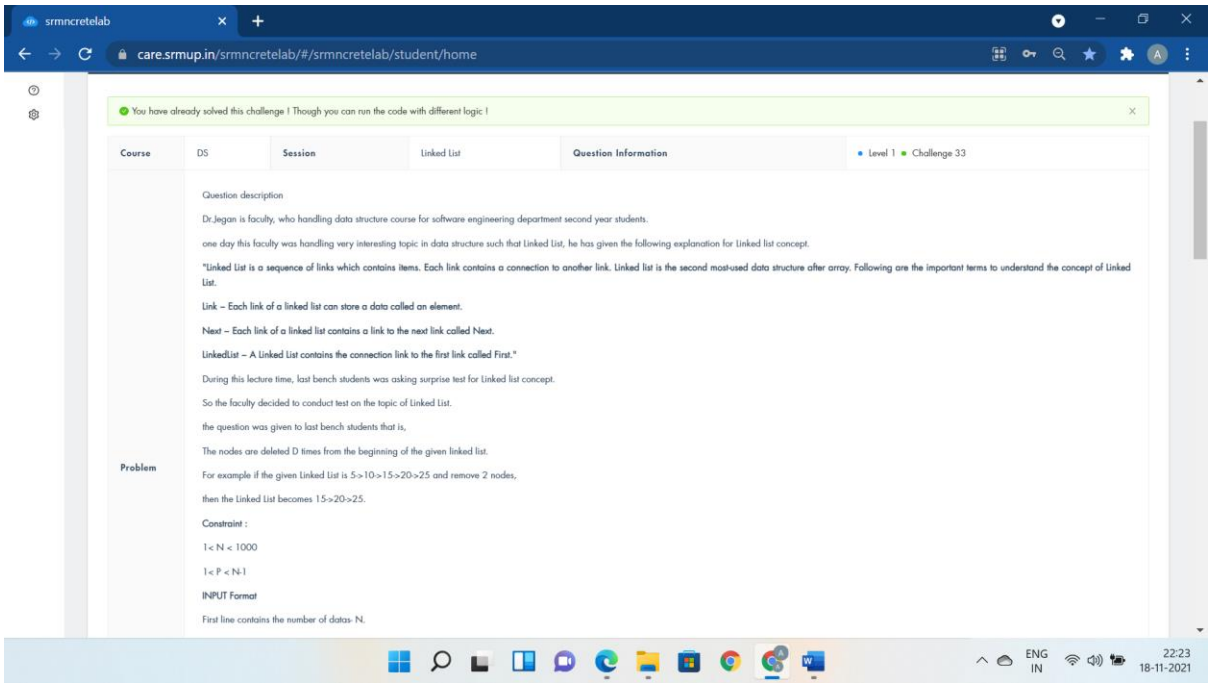
```
    create();
    return 0;
    cout << "for(i=0;i<n;i++)";
}
```




```
#include <bits/stdc++.h>
using namespace std;
struct node
{
        int key;
        struct node *next;
};
void push(struct node** head_ref, int new_key)
{
        struct node* new_node = new node();
        new_node->key = new_key;
        new_node->next = (*head_ref);
        (*head_ref) = new_node;
}
```

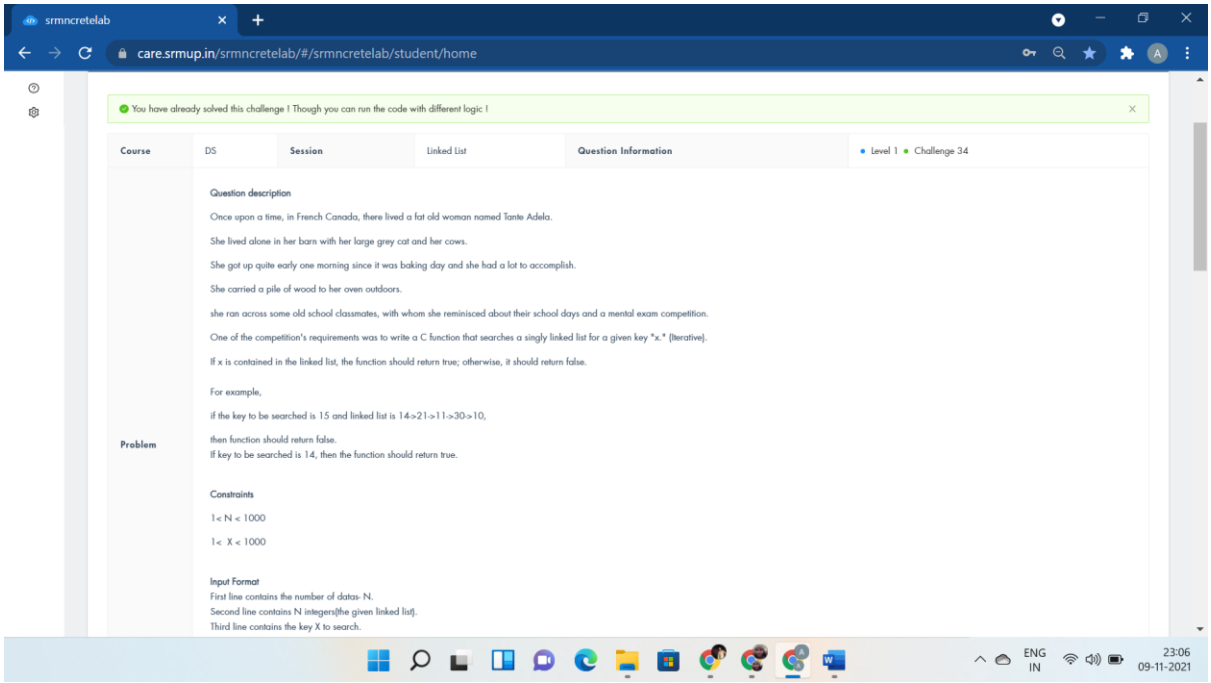
```
void printList(node *node){
	while (node != NULL)
	{
		cout<<"-->"<<node->key;
		node = node->next;
	}
}
int count(struct node* head,int search_for)
{
	node* current = head;
	int count=0;
	while (current != NULL)
	{
		if (current->key == search_for)
			count++;
		current = current->next;
	}
	return count;
}
int main()
{
	struct node* head = NULL;
	int x,n,t;
	cin>>n;
	while(n--){
	   cin>>t;
	   push(&head,t);
	}
	cin>>x;
	cout<<"Linked list:";
	printList(head);
	cout<<endl<<"Count of "<<x<<":"<<count(head, x);
	return 0;
```

}




```
#include<bits/stdc++.h>
using namespace std;
struct node {
   int data;
   node *next;
};
void insertAtEnd(node** head_ref, int new_data) {
   node* new_node = (node*)malloc(sizeof( node));
   node* last = *head_ref;
   new_node->data = new_data;
   new_node->next = NULL;
   if (*head_ref == NULL) {
      *head_ref = new_node;
      return;
   }
   while (last->next != NULL) last = last->next;
   last->next = new_node;
   return;
}
int main() {
```

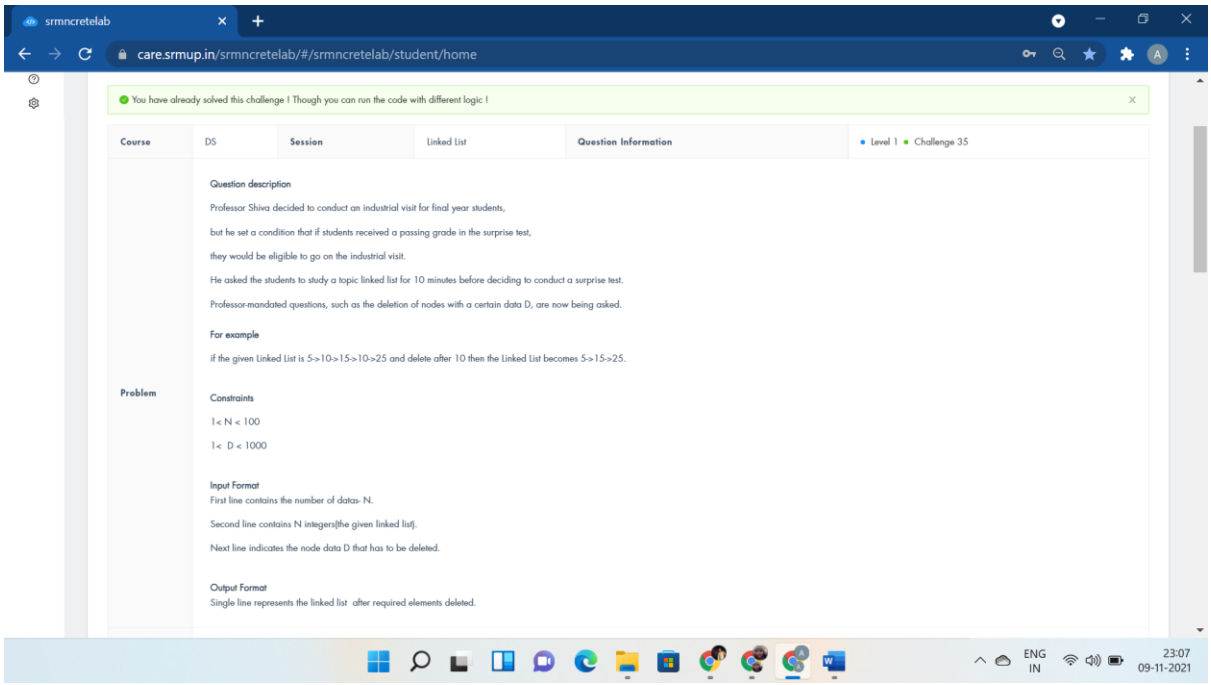
```
    node* head = NULL;
    int n,c,z,i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>c;
        insertAtEnd(&head,c);
    }
    cin>>z;
    for(int i=0;i<z;i++)
    head=head->next;
    cout << "Linked List:";
    node* node=head;
    while(node!=NULL){
        cout<<"->"<<node->data;
        node=node->next;
    }
    return 0;
    cout<<"void create()";
}
```




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

```
using namespace std;
struct node
{
	int key;
	struct node* next;
};
void push(struct node** head_ref, int new_key)
{
	struct node* new_node = new node();
	new_node->key = new_key;
	new_node->next = (*head_ref);
	(*head_ref) = new_node;
}
bool search(struct node* head,int x)
{
	node* current = head;
	while (current != NULL)
	{
		if (current->key == x)
			return true;
		current = current->next;
	}
	return false;
}
int main()
{
	struct node* head = NULL;
	int x,n,t;
	cin>>n;
	while(n--){
	   cin>>t;
	   push(&head,t);
	}
```

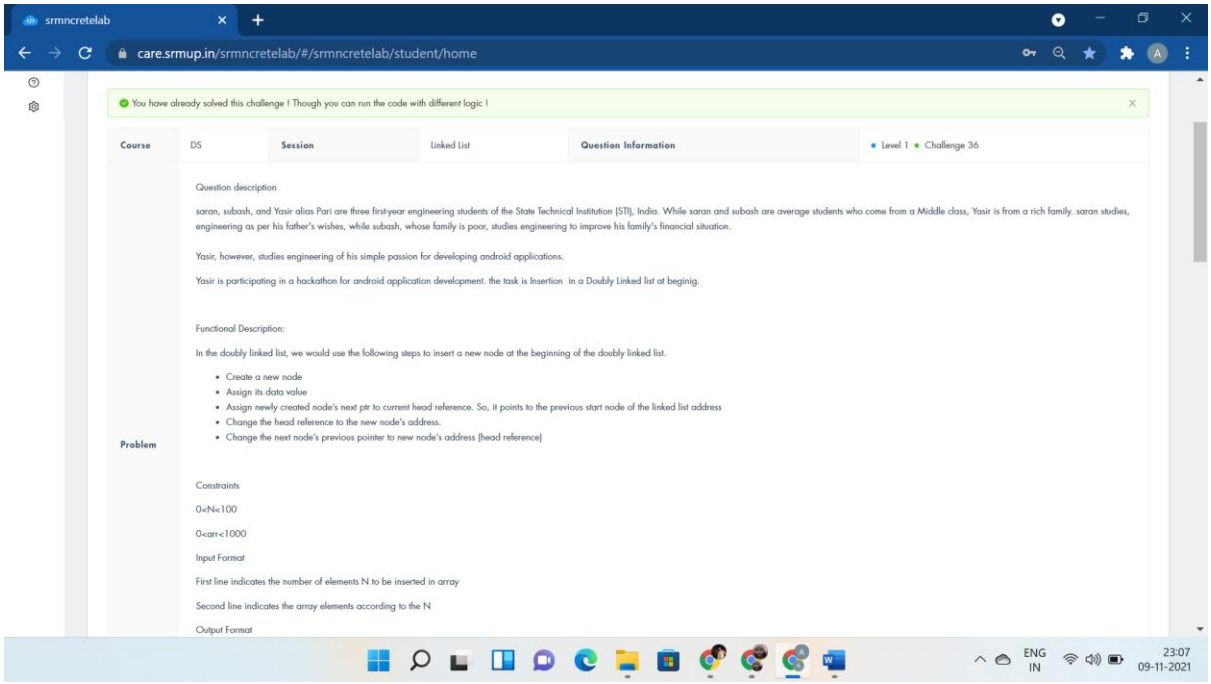
```
        cin>>x;
        search(head, x)? cout<<"Yes" : cout<<"No";
        return 0;
}
```




```
#include<iostream>
using namespace std;
struct node{
   int data;
   struct node *next;
}*start;
void display();
void deleteNode(node*& head, int val)
{
   if (head == NULL) {
      return;
   }
   if (head->data == val) {
      node* t = head;
      head = head->next;
```

```
        delete (t);
        return;
    }
    deleteNode(head->next, val);
}
int main() {
    int n;
    scanf("%d",&n);
    struct node *temp, *p2;
    start=NULL;
    for(int i=0;i<n;i++){
        temp=(struct node *)malloc(sizeof(struct node));
        scanf("%d", &temp -> data);
        temp->next = NULL;
        if(start == NULL){
            start= temp;
            p2 = temp;
        }
        else
        {
          p2->next=temp;
          p2=p2->next;
        }
    }
    int x;
    cin>>x;
    //display();
    for(int i=0;i<n;i++)
    deleteNode(start,x);
    display();
    return 0;
    cout<<"void del()void create() ";
}
```

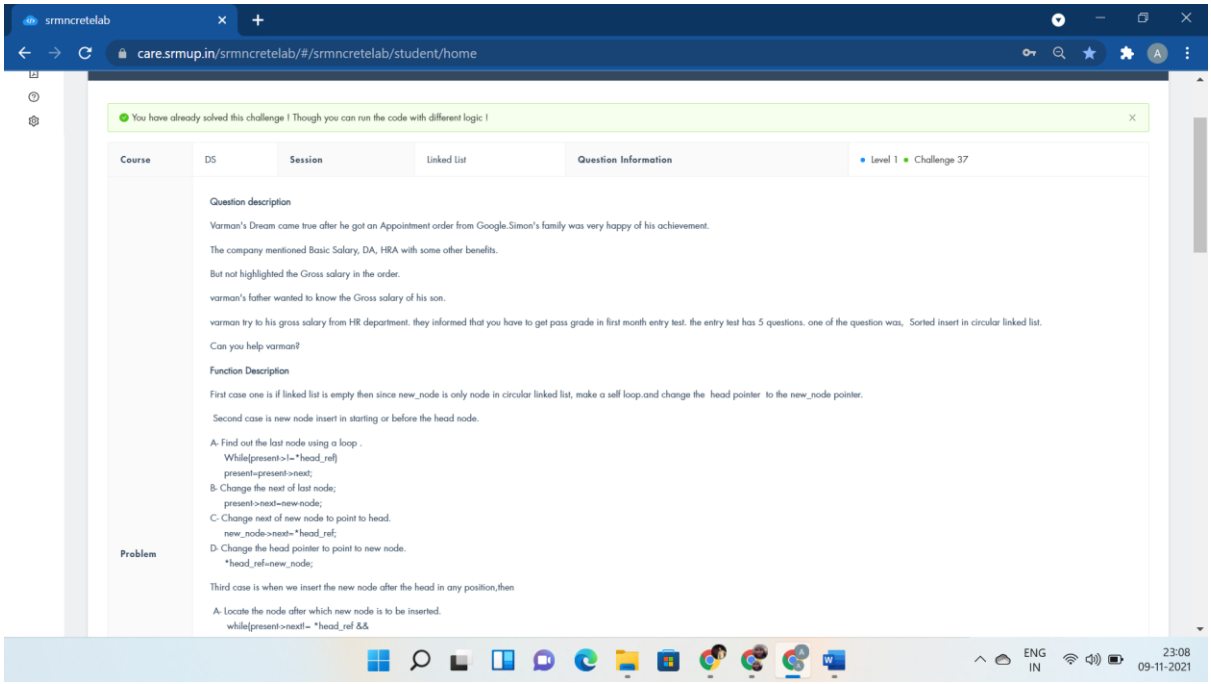
```
void display() {
   struct node *temp;
   temp = start;
   printf("Linked List:");
   while(temp != NULL)
   {
      printf("->%d",temp->data);
      temp = temp->next;
   }
}
```




```
#include <bits/stdc++.h>
using namespace std;
struct Node
{
        int data;
        struct Node *next;
        struct Node *prev;
};
```

```
void insertStart(struct Node** head,int data)
{
   struct Node* new_node = new Node();
   new_node->data = data;
   new_node->next = (*head);
   new_node->prev = NULL;
   if ((*head) != NULL)
      (*head)->prev = new_node;
   (*head) = new_node;
}
void printList(struct Node* node)
{
        Node* last;
        while (node != NULL)
        {
                cout<<node->data<<" ";
                last = node;
                node = node->next;
        }
        cout<<endl;
        while (last != NULL)
        {
                cout<<last->data<<" ";
                last = last->prev;
        }
}
int main()
{
        struct Node* head = NULL;
        int n;
        cin>>n;
        for(int i=0;i<n;i++){
           int t;
```

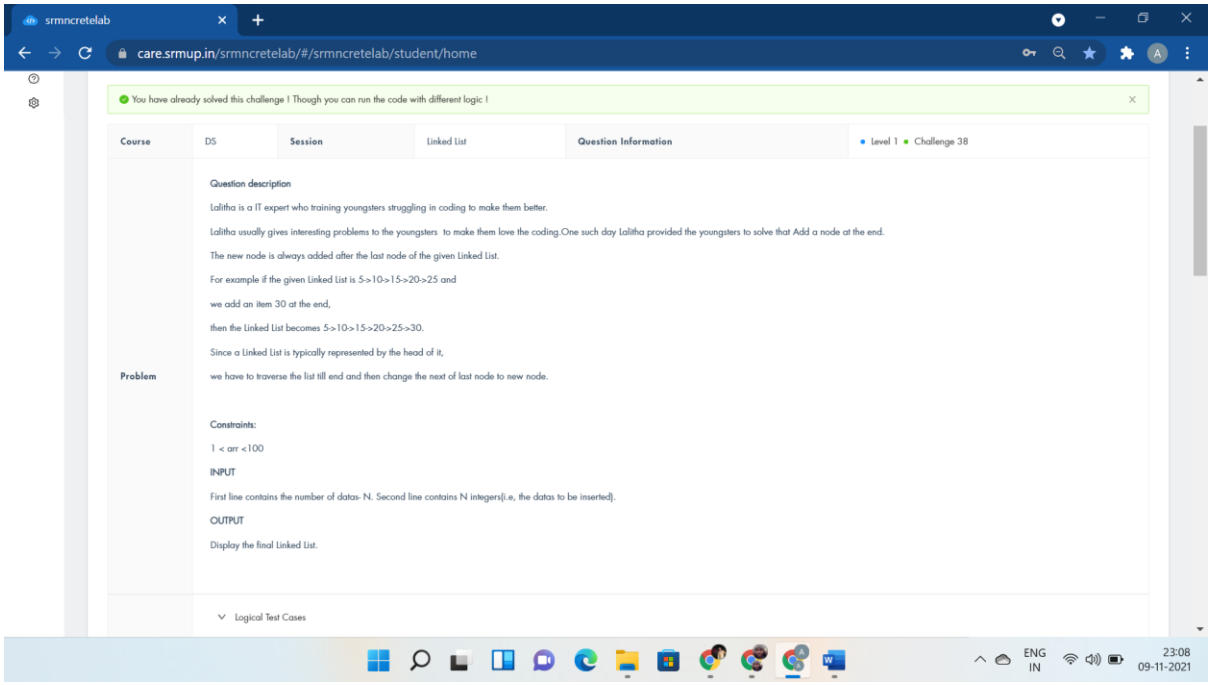
```
            cin>>t;
            insertStart(&head, t);
        }
        printList(head);
        return 0;
}
```




```
#include <stdio.h>
#include<stdlib.h>
struct Node{
    int data;
    struct Node *next;
};

void sortedInsert(struct Node** head_ref, struct Node* new_node)
{
    struct Node* current = *head_ref;

    if(current == NULL){
        new_node->next = new_node;
```

```
        *head_ref=new_node;
    }

    else if(current->data >= new_node->data){
        while(current->next != *head_ref)
        current=current->next;

        current->next=new_node;
        new_node->next=*head_ref;
        *head_ref=new_node;
    }

    else{
        while(current->next != *head_ref && current->next->data < new_node->data)
        current = current->next;

        new_node->next = current->next;
        current->next=new_node;
    }
}

void printList(struct Node *start){
    struct Node *temp;
    temp=start;

    do{
        printf("%d ",temp->data);
        temp=temp->next;
    }while(temp->next != start);
    printf("%d",temp->data);
}
int main()
{
```

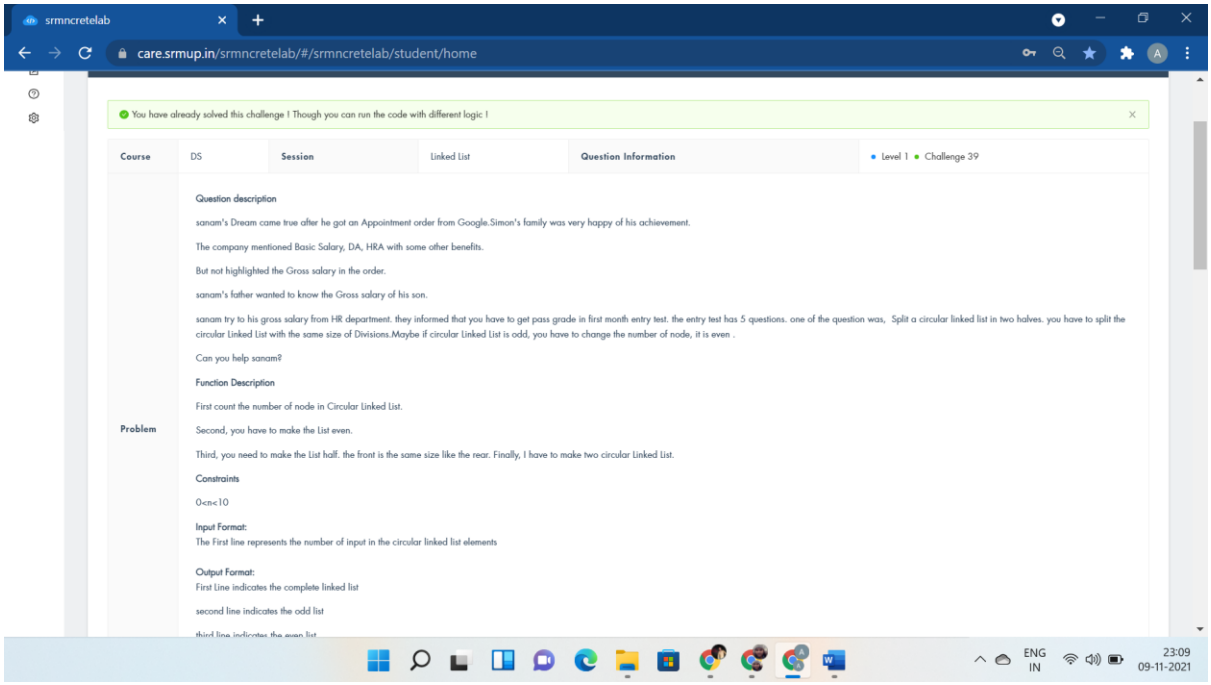
```
    int n,i;
    scanf("%d",&n);

    struct Node *start=NULL;
    struct Node *temp;

    for(i=0; i<n; i++){
        temp=(struct Node*)malloc(sizeof(struct Node));
        scanf("%d",&temp->data);
        sortedInsert(&start, temp);
    }
printList(start);
return 0;
}
```




```
#include <stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
```

```
}*start;
void display();
int main() {
   int n;
   scanf("%d",&n);
   struct node *temp, *p2;
   start=NULL;
  while(n) {
      temp=(struct node *)malloc(sizeof(struct node));
      scanf("%d", &temp -> data);
      temp->next = NULL;
      if(start == NULL){
         start= temp;
         p2 = temp;
      }
      else
      {
        p2->next=temp;
      //  while(p2 != NULL && p2 -> next != NULL       p2=p2->next;
        p2=p2->next;
      }--n;
   }
   display();
   return 0;
}
void display() {
   struct node *temp;
   temp = start;
   printf("Linked List:");
   while(temp != NULL)
   {
      printf("->%d",temp->data);
      temp = temp->next;
```

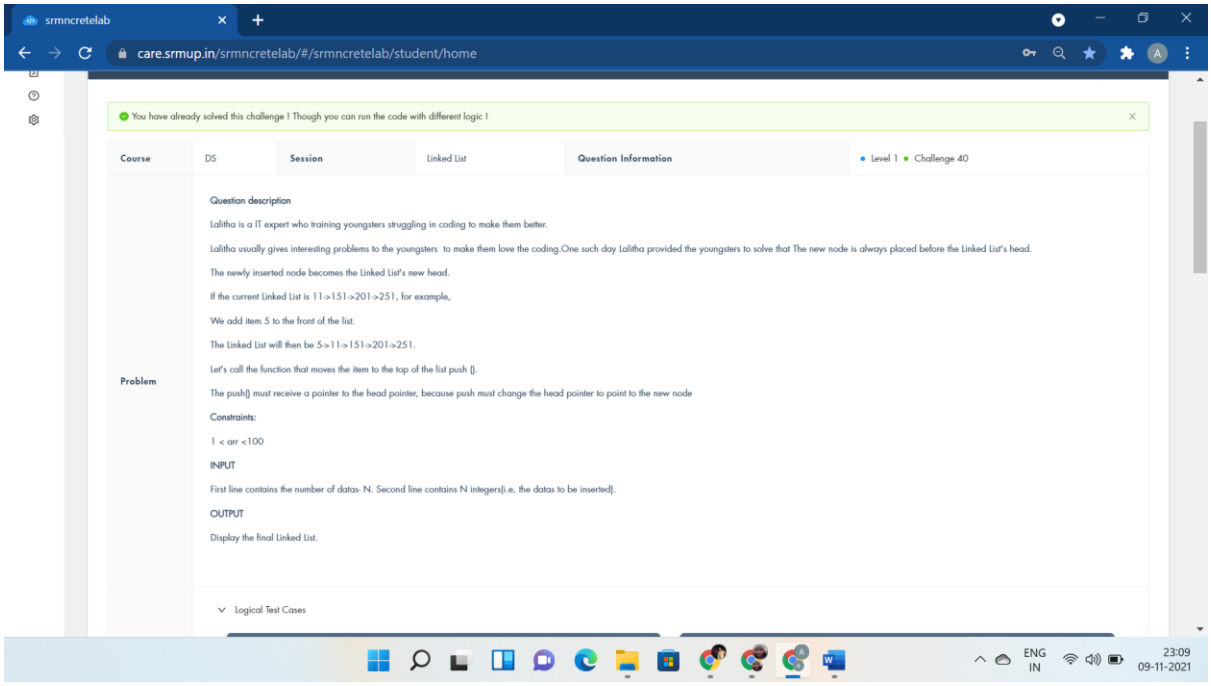
```
    }
}
```




```
#include <iostream>
using namespace std;
struct n
{
    int data;
    struct n *next;
} * odd, *even, *h = NULL, *tt;
void insert(int data)
{
    n *p = new n;
    p->data = data;
    p->next = NULL;
    tt->next = p;
    tt = p;
}
void oodd()
{
```

```
    cout << "Odd:\n";
    odd = h;
    int i = 1;
    cout << "[h]";
    while (odd != NULL)
    {
        if ((i % 2))
        {
            cout << "=>" << odd->data;
        }
        i++;
        odd = odd->next;
    }
    cout << "=>[h]";
}
void eeven()
{
    cout << "Even:\n";
    even = h;
    int i = 1;
    cout << "[h]";
    while (even != NULL)
    {
        if (!(i % 2))
        {
            cout << "=>" << even->data;
        }
        i++;
        even = even->next;
    }
    cout << "=>[h]";
}
void display(struct n *h)
```

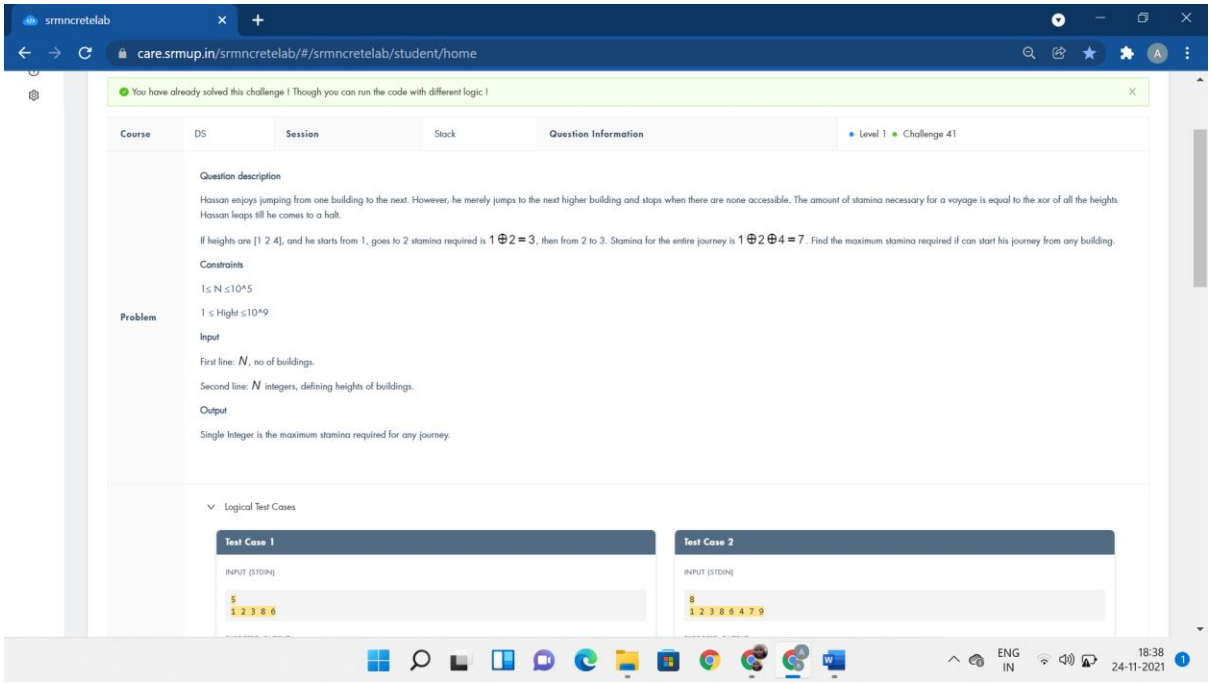
```
{
    cout << "Complete linked_list:\n[h]";
    while (h != NULL)
    {
        cout << "=>" << h->data;
        h = h->next;
    }
    cout << "=>[h]";
}
int main()
{
    int a;
    cin >> a;
    tt = new n;
    tt->data = 1;
    tt->next = NULL;
    h = tt;
    for (int i = 2; i <= a; i++)
    {
        insert(i);
    }
    n *y = h;
    display(y);
    cout << "\n";
    oodd();
    cout << "\n";
    eeven();
    return 0;
}
```




```
#include <bits/stdc++.h>
using namespace std;
struct node
{
	int data;
	node *next;
};
void push(node** start, int new_data){
	node* p1 = new node();
	p1->data = new_data;
	p1->next = *start;
	*start = p1;
}
void printList(node *node){
	while (node != NULL)
	{
		cout<<"->"<<node->data;
		node = node->next;
	}
}
```

```
int main(){
        node *start = NULL;
        int n,t;
        cin>>n;
        while(n--){
           cin>>t;
           push(&start,t);
        }
        cout<<"Linked List:";
        printList(start);
        return 0;
        cout<<"p1->next=start; void display()";
}
```

STACKS:-




```
#include <stdio.h>
int main() {
   int i, j, arr[1000000], n, temp=0,st[1000000]= {0};
   scanf("%d",&n);
```