Problem Statement 1

Alice is preparing for a coding competition and wants to practice her skills with linked lists. She decides to write a program that allows her to create a linked list by inserting elements at the front.

Help Alice with her task.

Company tags: HCL

Input format :

The first line of input consists of an integer **n,** representing the number of elements to be inserted.

The second line of input consists of **n** space-separated integers, representing the elements to be inserted in the linked list.

The third line of input consists of an integer **m**, representing the value to be inserted at the front of the linked list.

Output format :

The first line of output displays "Created Linked list: " followed by the elements of the linked list.

The second line of output displays "Final List: " followed by the elements of the linked list after inserting the new element at the front.

Refer to the sample output for formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 10$

$1 \leq$ elements, $m \leq 100$

Sample test cases :

Input 1 :

5

6 5 4 3 2

1

Output 1 :

Created Linked list: 2 3 4 5 6

Final List: 1 2 3 4 5 6

Input 2 :

4

-10 20 -30 40

5

Output 2 :

Created Linked list: 40 -30 20 -10

Final List: 5 40 -30 20 -10

```
#include <iostream>
using namespace std;

struct Node {
   int data;
   Node* next;

   Node(int val){
      data = val;
      next = nullptr;
   }
};

void insertAtFront(Node *&head,int val){
   Node *newNode = new Node(val);
   newNode->next = head;
   head = newNode;
}
```

```
void printList(Node *head){
    Node* temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp = temp->next;
    }
    cout<<endl;
}

int main() {
    int n;
    cin>>n;

    Node* head = nullptr;

    int val;

    for(int i=0; i<n; i++){
        cin>>val;
        insertAtFront(head, val);
    }

    cout<<"Created Linked list: ";
    printList(head);

    int m;
    cin>>m;
    insertAtFront(head, m);
```

```
    cout<<"Final List: ";
    printList(head);

    return 0;
}
```

Problem Statement 2

Kathir is a software engineer working on a project that involves managing a linked list of integers. His task is to develop a program that allows the user to insert integers at the beginning of the list and then display the entire list. The program should continue inserting elements until the user decides to stop(1).

Company tags: Wipro

Input format :

The first line of input consists of an integer n, representing the item to be inserted at the beginning of the linked list.

The second line of input consists of an integer (0 or 1), representing the choice to continue or stop the insertion process.

1. Input 0 to continue inserting more integers.
2. Input 1 to stop inserting and display the linked list.

Output format :

The output displays "Node inserted" each time a new integer is inserted into the linked list in a new line.

After insertion, the output displays "Linked List: " followed by the integers in the linked list in insertion order.

The final line of output displays "Node ended" after displaying the linked list.

Refer to the sample output for the formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 3500$

Sample test cases :

Input 1 :

2

0

4

1

Output 1 :

Node inserted

Node inserted

Linked List: 4 2

Node ended

Input 2 :

4

1

Output 2 :

Node inserted

Linked List: 4

Node ended

```
// You are using GCC
#include<iostream>
using namespace std;


struct Node {
   int data;
   Node * next;

   Node (int val){
      data = val;
```

```
        next = nullptr;
    }
};

void insertAtFront(Node*&head, int val){
    Node* newNode = new Node(val);
    newNode->next = head;
    head = newNode;
    cout<<"Node inserted"<<endl;
}

void printList(Node* head){
    cout<<"Linked List: ";
    Node* temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp  = temp -> next;
    }
    cout<<endl;
    cout<<"Node ended"<<endl;
}

int main() {
    Node* head = nullptr;

    int n, choice;

    do{
        cin>>n;
```

```
        insertAtFront(head, n);
        cin>>choice;
    }while(choice == 0);

    printList(head);

}
```

Problem statement 3

Fin is creating a linked list to store his favorite numbers in the order he inputs them. He wants to continue adding numbers until he inputs a negative number, which will signal the end of the input. After the input ends, he wants to print all the numbers in the linked list in the order they were added.

Company tags: Wipro

Input format :

The input consists of an integer n, representing the number to be added to the linked list in a new line each. The input ends when a negative number is entered.

Output format :

If the linked list is empty, the output displays the message "Linked List is empty".

If the linked list is not empty, the output displays "Linked List: " followed by the contents of the linked list separated by a space.

Each node's data is displayed in the order it was inserted.

Refer to the sample output for format specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$-5000 \leq n \leq 5000$

Sample test cases :

Input 1 :

1

2

3

4

-1

Output 1 :

Linked List: 1 2 3 4

Input 2 :

-2

Output 2 :

Linked List is empty.

Input 3 :

2000

-3000

4000

-5000

-1

Output 3 :

Linked List: 2000

```
// You are using GCC
#include<iostream>
using namespace std;

struct Node {
    int data;
    Node* next;

    Node(int val){
        data = val;
```

```
        next = nullptr;
    }
};

void insertAtEnd(Node*&head, int val){
    Node *newNode = new Node(val);
    if(head ==  nullptr){
        head = newNode;
    }else{
        Node *temp = head;
        while(temp->next != nullptr){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void printLL(Node* head){
    if(head == nullptr){
        cout<<"Linked List is empty.";
        return;
    }
    cout<<"Linked List: ";
    Node* temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp = temp->next;
    }
}
```

```
int main() {
    int n;
    Node *head = nullptr;



    while(true){
        cin>>n;
        if(n <0){
            break;
        }
        insertAtEnd(head, n);
    }


    printLL(head);


    return 0;
}
```

Problem Statement 4

Kamal wants to create a linked list and perform the following operations on it:

- Insert a node at the beginning of the linked list.
- After inserting, append a node at the end of the linked list.
- Print the final linked list.

Write a program that takes the number of nodes to be inserted, followed by their values, as input. After inserting the nodes, the program should ask for a new value and append a node with that value at the end of the linked list. Finally, the program should print the contents of the linked list.

Example

Input:

5

1 2 3 4 5

6

Output:

Created Linked list: 5 4 3 2 1

Final list: 5 4 3 2 1 6 // after appending the new value

Explanation:

The input specifies that there are 5 nodes, and their values are 1, 2, 3, 4, and 5 which are inserted at the beginning. So the created list is 5 4 3 2 1.

After inserting these nodes, the new value 6 is inserted at the end of the linked list. The final list is 5 4 3 2 1 6

Company tags: Capgemini

Input format :

The first line consists of an integer, **n**, representing the number of nodes to be initially inserted into the linked list.

The second line of input consists of **n** space-separated integers, representing the elements of the linked list.

The third line of input consists of an integer, **m**, representing the value of the new node to be inserted at the end of the linked list.

Output format :

The first line of output prints "Created Linked list: " followed by the initial linked list elements, separated by space.

The second line of output prints "Final list: " followed by the final linked list elements, after appending the new node, separated by space.

Refer to the sample output for formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 10$

$1 \leq$ elements, $m \leq 100$

Sample test cases :

Input 1 :

5

1 2 3 4 5

6

Output 1 :

Created Linked list: 5 4 3 2 1

Final list: 5 4 3 2 1 6

Input 2 :

3

10 20 30

40

Output 2 :

Created Linked list: 30 20 10

Final list: 30 20 10 40

```
// You are using GCC
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;

    Node(int val){
        data = val;
        next = nullptr;
```

```
    }
};

void push(Node** head_ref, int new_data)
{
    Node *newNode = new Node(new_data);
    newNode->next = *head_ref;
    *head_ref = newNode;
}


void append(Node** head_ref, int new_data)
{
    Node *newNode = new Node(new_data);
    if(*head_ref == nullptr){
        *head_ref = newNode;
    }else{
        Node *temp = *head_ref;
        while(temp->next != nullptr){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void printList(Node* node)
{
    Node *temp = node;
    while(temp != NULL){
```

```
        cout<<temp->data<<" ";
        temp = temp->next;
    }
}

int main()
{
    Node* head = NULL;

    int num_of_nodes, new_val;
    cin >> num_of_nodes;

    for (int i = 0; i < num_of_nodes; i++) {
        int val;
        cin >> val;
        push(&head, val);
    }

    cout << "Created Linked list:";
    printList(head);

    cin >> new_val;

    append(&head, new_val);

    cout << "\nFinal list:";
    printList(head);

    return 0;
```

}

Problem Statement 5

Ginny is building a linked list to store a sequence of numbers she inputs. After creating the linked list, she decides to insert a new value at a specific position in the list. She wants to print the final linked list after performing this insertion.

Input format :

The first line of input consists of an integer **n**, representing the number of nodes in the linked list.

The second line consists of **n** space-separated integers, representing the nodes.

The third line consists of an integer **m**, representing the position where the new element has to be added(1-indexed).

The fourth line consists of an integer **k**, representing the value of the element.

Output format :

The output displays the final linked list with the new value inserted at the specified position.

Refer to the sample output for the formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 40$

$1 \leq$ elements, $k \leq 2000$

$1 \leq m \leq n$

Sample test cases :

Input 1 :

5

2 3 9 6 1

3

8

Output 1 :

2 3 8 9 6 1

Input 2 :

10

8 3 12 77 30 55 94 28 56 14

8

99

Output 2 :

8 3 12 77 30 55 94 99 28 56 14

```
// You are using GCC
#include<iostream>

using namespace std;

struct Node{
    int data;
    Node* next;

    Node(int val){
        data = val;
        next = nullptr;
    }
};

void insertAtEnd(Node*& head, int val){
    Node *newNode = new Node(val);
    if(head == nullptr){
        head = newNode;
    }else{
        Node* temp = head;
```

```
        while(temp->next != nullptr){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void insertAtPosition(Node*& head, int index, int num){
    Node *newNode = new Node(num);

    if(index == 1){
        newNode->next = head;
        head = newNode;
        return;
    }

    Node *temp = head;
    for(int i=1; i< index-1 && temp != nullptr; i++){
        temp = temp->next;
    }

    if(temp){
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void printLL(Node *head){
    Node *temp = head;
```

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

int main() {
    int n;
    cin>>n;

    Node* head = nullptr;

    for(int i=0; i<n; i++){
        int val;
        cin>>val;
        insertAtEnd(head, val);
    }

    int index;
    cin>>index;


    int num;
    cin>>num;

    insertAtPosition(head, index, num);

    printLL(head);
```

```
    return 0;
}
```

Problem Statement 6

Lisa wants to create a linked list sorted in ascending order. She wants to insert nodes in such a way that the linked list remains sorted after insertion.

Write a program that takes the number of nodes to be inserted, followed by their values in non-decreasing order. The program should then ask for a new value and insert a node with that value at the appropriate position to maintain the sorted order.

Finally, the program should print the updated linked list.

Example

Input:

5

1 3 5 7 9

4

Output:

1 3 4 5 7 9

Explanation:

The program first creates a sorted linked list using the given input values: 1, 3, 5, 7, and 9. After creating the initial sorted linked list, the program asks for a new value, which is 4. It then inserts a node with value 4 at the appropriate position to maintain the sorted order. Finally, the program prints the updated linked list, which is 1, 3, 4, 5, 7, and 9 in ascending order.

Company tags: Wipro

Input format :

The first line of input consists of an integer **n**, representing the number of elements in the initial sorted linked list.

The second line of input consists of **n** space-separated integers, representing the elements of the sorted linked list.

The third line of input consists of integer **data**, which represents the element to be inserted into the linked list.

Output format :

The output displays the updated linked list after inserting the new element, separated by space.

The linked list should remain sorted in ascending order.

Refer to the sample output for the formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 10$

$-100 \leq$ elements, data ≤ 100

Sample test cases :

Input 1 :

5

1 3 5 7 9

4

Output 1 :

1 3 4 5 7 9

Input 2 :

6

-15 -10 0 5 9 10

7

Output 2 :

-15 -10 0 5 7 9 10

```
#include <iostream>
using namespace std;

struct Node {
   int data;
   Node* next;

   Node(int val){
      data = val;
      next = nullptr;
   }
};

void sortedInsert(Node** head_ref, int new_data)
{
   Node *newNode = new Node(new_data);

   //If the list is empty or the node should be inserted at the beginning
   if( *head_ref == NULL || new_data < (*head_ref)->data){
      newNode->next = *head_ref;
      *head_ref = newNode;
      return;
   }

   //Locate the node after which the new Node is to be inserted
   Node *current = *head_ref;
   while(current->next != NULL && current->next->data < new_data){
      current = current->next;
   }
```

```
    newNode->next = current->next;
    current->next = newNode;
}

void printList(Node* head)
{
    Node *temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp = temp->next;
    }

}

int main()
{
    Node* head = NULL;
    int n, data;

    cin >> n;

    for (int i = 0; i < n; i++) {
        cin >> data;
        sortedInsert(&head, data);
    }

    cin >> data;
    sortedInsert(&head, data);
```

```
    printList(head);

    return 0;
}
```

Problem Statement 7

Vijay is creating a linked list and wants to perform different operations on it. She can append values either to the left (beginning) or right (end) of the linked list. She also wants to print the linked list at any point in time.

He wants to be able to perform the following operations:

1: Append Left: Append a node at the beginning(left) of the linked list.

2: Append Right: Append a node at the end(right) of the linked list.

3: Print: Print the contents of the linked list.

4: Exit: Exit the program.

Company tags: Accenture

Input format :

The input consists of multiple lines. Each line starts with an integer choice representing the operation (1 to append left, 2 to append right, 3 to print, and 4 to exit).

If the choice is 1 or 2, it is followed by an integer **n** representing the value to be appended to the linked list separated by a space.

The input continues until the choice is 4.

Output format :

When the choice is 3, the output displays "Linked List: " followed by the values in the linked list separated by a space.

If the choice is invalid the output displays "Invalid choice".

Refer to the sample output for the formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 100$

Sample test cases :

Input 1 :

1 10

2 20

3

4

Output 1 :

Linked List: 10 20

Input 2 :

1 10

1 20

1 30

2 35

3

4

Output 2 :

Linked List: 30 20 10 35

Input 3 :

5

4

Output 3 :

Invalid choice

```
// You are using GCC
#include <iostream>
using namespace std;
```

```
struct Node {
    int val;
    Node* next;

    Node(int data){
        val = data;
        next = nullptr;
    }
};

void appendLeft(Node** head, int val) {
    Node* newNode = new Node(val);
    newNode->next = *head;
    *head = newNode;
}

void appendRight(Node** head, int val) {
    Node *newNode = new Node(val);
    if(*head == nullptr){
        *head = newNode;
    }else{
        Node* temp = *head;
        while(temp->next != NULL){
            temp = temp->next;
        }
        temp->next = newNode;
    }
```

```
}

void print(Node* head) {
    Node *temp = head;
    while(temp != NULL){
        cout<<temp->val<<" ";
        temp = temp->next;
    }

}

int main() {
    Node* myList = nullptr;
    int choice;
    int val;

    do {
    cin >> choice;


    switch (choice) {
        case 1:
            cin >> val;
            appendLeft(&myList, val);
            break;
        case 2:
            cin >> val;
            appendRight(&myList, val);
            break;
```

```
        case 3:
            cout << "Linked List: ";
            print(myList);
            break;
        case 4:
            break;
        default:
            cout << "Invalid choice" ;
    }

} while (choice != 4);


    return 0;
}
```

Problem Statement 8

Yamini is creating a linked list to store a sequence of numbers she inputs. She wants to input a number of elements and add them to the end of the linked list. After all elements are added, she wants to print the entire linked list. Yamini should note that the code handles an empty list case and prints a message if the list is empty.

Company tags: Accenture

Input format :

The first line of input consists of an integer **n**, representing the number of elements to be added to the linked list.

The second line of input consists of **n** space-separated integers, representing the elements to be added to the linked list.

Output format :

The output displays the elements of the linked list in the order they were added, separated by spaces.

If the linked list is empty, the program will output "The list is empty"

Refer to the sample output for formatting specifications.

Code constraints :

In this scenario, the test cases fall under the following constraints:

$1 \leq n \leq 20$

$0 \leq$ elements ≤ 100

Sample test cases :

Input 1 :

4

7 8 6 4

Output 1 :

7 8 6 4

Input 2 :

0

Output 2 :

The list is empty

```
// You are using GCC
#include<iostream>
using namespace std;

struct Node {
   int data;
   Node* next;

   Node(int val){
      data = val;
      next = nullptr;
```

```
    }
};

void insertAtEnd(Node*&head, int val){
    Node *newNode = new Node(val);
    if(head == nullptr){
        head = newNode;
    }else{
        Node* temp = head;
        while(temp->next != NULL){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void printLL(Node *head){
    Node* temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp = temp->next;
    }
}

int main() {
    int n;
    cin>>n;

    if(n == 0)
```

```
    {
        cout<<"The list is empty";
    }

    Node* head = nullptr;

    for(int i=0; i<n; i++){
        int val;
        cin>>val;
        insertAtEnd(head, val);
    }

    printLL(head);

    return 0;
}
```

Problem Statement 9

A teacher wants to maintain a record of student grades for a class test using a singly linked list. Initially, the grades inserted at front, when the students submitted their tests. Later, the teacher decides to add an extra credit assignment score for a new student at the front of the linked list.

Assist the teacher in printing the updated linked list of grades.

Input format :

The first line of input consists of an integer **n,** representing the number of students.

The second line consists of **n** space-separated integers, representing the grades.

The third line consists of an integer representing the grade value to be inserted at the front.

Output format :

The output prints the linked list elements, after inserting the given grade at the front, separated by space.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 10$

$1 \leq grade \leq 100$

Sample test cases :

Input 1 :

5

36 45 74 83 26

91

Output 1 :

91 26 83 74 45 36

Input 2 :

4

71 42 63 75

52

Output 2 :

52 75 63 42 71

```
// You are using GCC
#include<iostream>
using namespace std;

struct Node{
    int data;
    Node* next;

    Node(int val){
```

```
        data = val;
        next = nullptr;
    }
};

void insertAtFront(Node*&head, int val){
    Node* newNode = new Node(val);
    newNode->next = head;
    head = newNode;
}

void printLL(Node* head){
    Node* temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp = temp->next;
    }
}

int main() {
    int n;
    cin>>n;
    Node *head = nullptr;

    for(int i=0; i<n; i++){
        int val;
        cin>>val;
        insertAtFront(head, val);
    }
```

```
    int m;
    cin>>m;
    insertAtFront(head, m);

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

Problem Statement 10

Sarath is developing a text document editor that supports adding text lines at both the beginning and the end of the document. The editor should first read a list of initial lines to insert at the beginning of the document and then read an additional line to append at the end.

Guide Sarath in implementing the functionality to handle these operations using a singly linked list and display the final state of the document.

Company tags: Capgemini

Input format :

The first line of input consists of an integer, **n** representing the number of lines in the document.

The next **n** lines consist of the lines of text that constitute the document.

The last line of input consists of a string **s**, which needs to be appended at the end of the document.

Output format :

The first line of output prints "Document: " followed by the initial document content, which inserts the given **n** values at the beginning.

The second line prints "Updated Document: " followed by the final document content, which appends the given value **s** at the end of the document.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq n \leq 10$

Sample test cases :

Input 1 :

3

Apple

Banana

Orange

Grapes

Output 1 :

Document: Orange Banana Apple

Updated Document: Orange Banana Apple Grapes

Input 2 :

2

Hello

World

Space

Output 2 :

Document: World Hello

Updated Document: World Hello Space

```
// You are using GCC
#include<iostream>
#include<string>
using namespace std;

struct Node{
    string data;
```

```
    Node* next;

    Node(string val){
        data = val;
        next = nullptr;
    }
};

void insertAtFront(Node*& head, string val){
    Node* newNode = new Node(val);
    newNode->next = head;
    head = newNode;
}

void insertAtEnd(Node*& head, string val){
    Node* newNode = new Node(val);
    if(head == nullptr){
        head = newNode;
    }else{
        Node* temp = head;
        while(temp->next != NULL){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void printLL(Node * head){
    Node* temp = head;
```

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

int main() {
    int n;
    cin>>n;
    cin.ignore();
    Node *head = nullptr;

    for(int i=0; i<n; i++){
        string line;
        getline(cin, line);
        insertAtFront(head, line);
    }

    cout<<"Document: ";
    printLL(head);
    cout<<endl;

    string s;
    getline(cin, s);

    insertAtEnd(head, s);

    cout<<"Updated Document: ";
    printLL(head);
```

```
    return 0;

}
```

Problem Statement 11

Imagine you are developing a contact management application where users can maintain a singly linked list of names.

Initially, users can enter a sequence of names to form a list of contacts. Later, they can insert a new name at a specific position within the list to rearrange their contacts, ensuring the list is updated and displayed correctly.

Input format :

The first line of input consists of an integer **n**, representing the number of initial names.

The next **n** lines each contain a single name, representing the initial list of contacts.

The last line consists of a name and an integer **pos,** representing the new contact to be added and its position in the list.

Output format :

The output prints the updated list of contacts after inserting the new contact at the specified position.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq n \leq 10$

1 ≤ length of each name ≤ 100

Sample test cases :

Input 1 :

4

John

Alice

Bob

Emma

Michael 3

Output 1 :

John Alice Michael Bob Emma

Input 2 :

3

Emma

Daniel

Sophia

Charlotte 1

Output 2 :

Charlotte Emma Daniel Sophia

```
// You are using GCC
#include<iostream>
#include<string>
using namespace std;

struct Node{
   string data;
   Node* next;

   Node(string val){
      data = val;
      next = nullptr;
   }
};
```

```
void insertAtEnd(Node*&head, string val){
    Node* newNode = new Node(val);
    if(head == nullptr){
        head = newNode;
    }else{
        Node* temp = head;
        while(temp->next != NULL){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void insertAtPosition(Node*&head, int index, string value){
    Node* newNode = new Node(value);

    if(index == 1){
        newNode->next = head;
        head = newNode;
        return;
    }

    Node* temp = head;
    for(int i=1; i<index-1 && temp!= nullptr; i++){
        temp = temp->next;
    }

    if(temp){
    newNode->next = temp->next;
```

```
    temp->next = newNode;
    }
}


void printLL(Node*&head){
    Node*temp = head;

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


int main() {
    int n;
    cin>>n;
    cin.ignore();
    Node* head = nullptr;

    for(int i=0; i<n; i++){
        string name;
        getline(cin, name);
        insertAtEnd(head, name);
    }

    string newname;
    int pos;
```

```
    cin>>newname;
    cin>> pos;

    insertAtPosition(head, pos, newname);

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

Problem Statement 12

Sharon is developing a character editor for a simple text-based game. The editor allows users to maintain a singly linked list of characters representing a sequence of moves in the game.

Initially, users input a series of characters to create their move sequence. Later, they can insert a new character at a specific position within the sequence to alter their gameplay strategy, ensuring the list is updated and displayed correctly. Assist Sharon in the task.

Company tags: TCS

Input format :

The first line contains an integer **n,** the number of initial characters.

The second line consists of **n** characters, representing the initial move sequence.

The last line contains a character and an integer **pos,** representing the new move to be added and its position(1-based) in the sequence.

Output format :

The first line displays "Current Linked List:" followed by the initial sequence of moves in the next line.

The third line displays "Updated Linked List:" followed by the updated sequence of moves after inserting the new move at the specified position in the next line.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq n \leq 10$

Sample test cases :

Input 1 :

6

A B C D E F

G 7

Output 1 :

Current Linked List:

A B C D E F

Updated Linked List:

A B C D E F G

Input 2 :

3

A O B

M 1

Output 2 :

Current Linked List:

A O B

Updated Linked List:

M A O B

```
// You are using GCC
#include<iostream>
using namespace std;

struct Node{
    char data;
```

```
    Node* next;

    Node(char val){
        data = val;
        next = nullptr;
    }
};

void insertAtEnd(Node*&head, char value){
    Node* newNode = new Node(value);
    if(head == nullptr){
        head = newNode;
    }else{
        Node* temp = head;
        while(temp->next != nullptr){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void insertAtPosition(Node*&head, int index, char value){
    Node* newNode = new Node(value);

    if(index == 1){
        newNode->next = head;
        head = newNode;
        return;
    }
```

```
    Node *temp = head;
    for(int i=1; i<index-1 && temp != nullptr; i++){
        temp = temp->next;
    }

    if(temp){
        newNode->next = temp->next;
        temp->next = newNode;
    }
}

void printLL(Node*&head){
    Node* temp = head;
    while(temp != NULL){
        cout<<temp->data<<" ";
        temp = temp->next;
    }
}

int main(){
    int n;
    cin>>n;
    cin.ignore();
    Node *head = nullptr;

    char alpha;

    for(int i=0; i<n; i++){
```

```
        cin>>alpha;
        insertAtEnd(head, alpha);
    }

    cout<<"Current Linked List:"<<endl;
    printLL(head);

    char newchar;
    int pos;
    cin>>newchar;
    cin>>pos;

    insertAtPosition(head, pos, newchar);

    cout<<"Updated Linked List:"<<endl;
    printLL(head);
    return 0;
}
```