Problem Statement 1

A communication device transmits a binary sequence through a singly linked list, where each node represents a binary digit. Your task is to read the binary digits, construct the linked list, and determine the decimal value of the transmitted binary sequence.

Company Tags: Infosys

Input format :

The first line of input consists of an integer **n,** the number of binary digits.

The second line consists of **n** binary digits (0 or 1) separated by spaces.

Output format :

The first line of output prints the binary sequence as a linked list.

The second line prints the decimal value of the binary sequence.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 15$

The binary digits will be 0 or 1.

Sample test cases :

Input 1 :

3

1 0 1

Output 1 :

Linked List: 1 0 1

Decimal Value: 5

Input 2 :

4

1 0 1 0

Output 2 :

Linked List: 1 0 1 0

Decimal Value: 10

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

struct Node{
    int data;
    Node* next;

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

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

    if(head == nullptr){
        head = newNode;
    }else{
        Node* temp = head;
        while(temp->next != nullptr){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

int convertToDec(Node*& head){
    Node* temp = head;
    int result = 0;
```

```
    while(temp != NULL){
        result = result*2 + temp->data;
        temp = temp->next;
    }

    return result;
}

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

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

    int num;
    Node* head = nullptr;

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

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

```
    cout<<"Decimal Value: ";
    cout<<convertToDec(head)<<endl;

    return 0;
}
```

Problem Statement 2

In a warehouse management system, packages are organized sequentially using a singly linked list, where each node represents a package with its unique ID. To optimize the retrieval process, the system allows reversing the order of packages in groups of k nodes.

Your task is to read the package IDs, construct the linked list, and reverse the order of packages in groups of k.

Input format :

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

The second line consists of **n** integers, the package IDs.

The third line consists of an integer **k,** the size of the groups to reverse.

Output format :

The first line of output prints the original sequence of package IDs as a linked list.

The second line prints the modified sequence of package IDs after reversing in groups of k.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 20$

$1 \leq$ package ID, $k \leq 100$

Sample test cases :

Input 1 :

8

1 2 3 4 5 6 7 8

4

Output 1 :

Original Linked List: 1 2 3 4 5 6 7 8

Modified Linked List: 4 3 2 1 8 7 6 5

Input 2 :

5

1 2 3 4 5

3

Output 2 :

Original Linked List: 1 2 3 4 5

Modified Linked List: 3 2 1 5 4

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

struct Node {
    int data;
    Node* next;
};

Node* createNode(int data) {

    Node *newNode = new Node;
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;

}

Node* reverseKNodes(Node* head, int k) {

    Node *curr = head;
```

```
    Node *prev = NULL;
    Node *next = NULL;

    int count =0;

    while(curr != nullptr && count < k){
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
        count++;
    }

    if(next != nullptr){
        head->next = reverseKNodes(next, k);
    }

    return prev;
}

void displayList(Node* head) {

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

void deleteList(Node* head) {
    Node* temp;
```

```
    while (head != NULL) {
        temp = head;
        head = head->next;
        delete temp;
    }
}

int main() {
    int n, k;
    cin >> n;
    Node* head = NULL;
    Node* tail = NULL;

    for (int i = 0; i < n; i++) {
        int value;
        cin >> value;
        Node* newNode = createNode(value);
        if (head == NULL) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
    }

    cin >> k;

    cout << "Original Linked List: ";
    displayList(head);
    cout<<endl;
```

```
    head = reverseKNodes(head, k);

    cout << "Modified Linked List: ";
    displayList(head);

    deleteList(head);

    return 0;
}
```

Problem Statement 3

Sarath is working on a sales data management system. The sales data for each day is represented by two singly linked lists, where each node contains the sales values. The task is to calculate the total sales by adding the sales values from the two linked lists and returning the result in a new linked list.

Can you assist Sarath in the program?

Suppose the first linked list is 6 3 and the second linked list is 7,

1. First Node Addition: 3 (from the first list) + 7 (from the second list) = 10
2. Second Node Addition:6 (from the first list) + 1 (carry) = 7

So, the result is 7 0.

Company Tags: Google, Amazon

Input format :

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

The second line consists of **N** space-separated integers, representing the sales values for each node in the first linked list.

The third line consists of an integer **M**, representing the number of nodes in the second linked list.

The fourth line consists of **M** space-separated integers, representing the sales values for each node in the second linked list.

Output format :

The first line of output prints the sales values of the first linked list, separated by space.

The second line prints the sales values of the second linked list, separated by space.

The third line prints the total sales, separated by space.

Refer to the sample output for formatting specifications.

Code constraints :

The size of the first linked list (N) and the size of the second linked list (M) are non-negative integers.

Sample test cases :

Input 1 :

3

4 5 1

3

3 4 5

Output 1 :

First linked list: 4 5 1

Second linked list: 3 4 5

Total Sales: 7 9 6

Input 2 :

2

6 3

1

7

Output 2 :

First linked list: 6 3

Second linked list: 7

Total Sales: 7 0

Input 3 :

3

1 3 5

3

2 3 4

Output 3 :

First linked list: 1 3 5

Second linked list: 2 3 4

Total Sales: 3 6 9

```
// 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;
    }
}
```

```
Node* reverse(Node* head){
   Node* next = NULL;
   Node* curr = head;
   Node* prev = NULL;
   while(curr){
      next = curr->next;
      curr->next = prev;
      prev = curr;
      curr = next;
   }
   return prev;
}

Node* addLists(Node* l1, Node* l2){
   l1 = reverse(l1);
   l2 = reverse(l2);

   Node* result = nullptr;
   int carry =0;

   while(l1 || l2 || carry){
      int sum = carry;

      if(l1){
         sum += l1->data;
         l1 = l1->next;
      }

      if(l2){
         sum += l2->data;
```

```
            l2 = l2->next;
        }

        carry = sum/10;
        insertAtEnd(result, sum%10);
    }

    return reverse(result);
}

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

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

    Node* first = nullptr;

    for(int i=0; i<n; i++){
        int l1;
        cin>>l1;
        insertAtEnd(first, l1);
    }
```

```
    cout<<"First linked list: ";
    printLL(first);

    int m;
    cin>>m;

    Node* second = nullptr;

    for(int i=0; i<m; i++){
        int l2;
        cin>>l2;
        insertAtEnd(second, l2);
    }

    cout<<"Second linked list: ";
    printLL(second);


    Node* total = addLists(first, second);

    cout<<"Total Sales: ";
    printLL(total);

    return 0;

}
```

Problem Statement 4

Prasath is developing a program to manage sorted singly linked lists of sales data. The linked list should be kept sorted in ascending order. The task is to insert a new data point into the sorted linked list and display the list before and after insertion.

Help Prasath in creating the program.

Company Tags: Capgemini

Input format :

The first line of input contains an integer **n,** the number of initial data points to be inserted into the linked list.

The second line contains **n** integers, which are the initial data points.

The third line contains a single integer **data,** the new data point to be inserted into the linked list.

Output format :

The first line of output prints "Original data points:" followed by the linked list elements before the insertion of the new data point.

The second line prints "Updated data points:" followed by the linked list elements after the insertion of the new data point.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 25$

1 ≤ initial data points, data ≤ 100

Sample test cases :

Input 1 :

6

25 36 47 58 69 80

19

Output 1 :

Original data points: 25 36 47 58 69 80

Updated data points: 19 25 36 47 58 69 80

Input 2 :

2

50 100

75

Output 2 :

Original data points: 50 100

Updated data points: 50 75 100

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

struct Node{
    int data;
    Node* next;

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



void insertSorted(Node*&head, int val){
    Node* newNode = new Node(val);

    if(!head || val < head->data){
        newNode->next = head;
        head = newNode;
        return;
    }
```

```
    Node* temp = head;
    while(temp->next != nullptr && temp->next->data < val){
        temp = temp->next;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

void printLL(Node* head){
    Node* temp = head;
    while(temp != nullptr){
        cout<<temp->data<<" ";
        temp = temp->next;
    }
}
int main() {
    int n;
    cin>>n;

    int num;
    Node* head = nullptr;

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

    cout<<"Original data points: ";
    printLL(head);
    cout<<endl;
```

```
    int val;
    cin>>val;
    insertSorted(head, val);

    cout<<"Updated data points: ";
    printLL(head);

    return 0;
}
```

Problem Statement 5

Imagine you are working on a program for an inventory management system in a retail store. The store uses barcodes to label and track its products. Each barcode is represented as a singly linked list, where each digit of the barcode is stored in a separate node. Your task is to write a program that adds 1 to the barcode value and updates the linked list accordingly.

For example, 1999 is represented as (1-> 9-> 9 -> 9), and adding 1 to it should change it to (2->0->0->0).

Company Tags: Flipkart

Input format :

The first line of input contains an integer **n,** indicating the number of digits in the barcode.

The second line contains **n** space-separated integers, representing the digits of the barcode.

Output format :

The output prints the linked list representing the modified barcode, after adding 1 to its value.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 20$

$0 \leq$ barcode digits ≤ 9

Sample test cases :

Input 1 :

4

1 9 9 9

Output 1 :

2 0 0 0

Input 2 :

6

1 2 3 4 5 6

Output 2 :

1 2 3 4 5 7

```
// 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;
    }
}

Node* reverse(Node* head){
    Node* prev = nullptr;
    Node* curr = head;
    Node* next = nullptr;

    while(curr){
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}

Node* add1ToLL(Node* l){
    l = reverse(l);

    Node* result = nullptr;
    int carry =1;

    while(l || carry){
        int sum = carry;
```

```
        if(l){
            sum += l->data;
            l = l->next;
        }

        carry = sum/10;
        insertAtEnd(result, sum%10);
    }

    return reverse(result);
}

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

```
    Node* total = add1ToLL(head);

    printLL(total);

    return 0;
}
```

Problem Statement 6

You are tasked with designing a program that operates on two singly linked lists. Your objective is to create a new linked list that represents the union of the two given linked lists while ensuring that the elements in the union list are distinct and sorted in ascending order.

Company Tags: Microsoft

Input format :

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

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

The third line consists of an integer **m**, representing the number of nodes in the second linked list.

The fourth line consists of **m** space-separated integers, representing the nodes in the second linked list.

Output format :

The first line of output displays the nodes of the first linked list, sorted in ascending order.

The second line displays the nodes of the second linked list, sorted in ascending order.

The third line displays the nodes of the union linked list after merging the distinct elements from the first and second linked lists.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n, m \leq 15$

$0 \leq$ nodes ≤ 100

Sample test cases :

Input 1 :

6

9 6 4 2 3 8

5

1 2 8 6 2

Output 1 :

First Linked List: 2 3 4 6 8 9

Second Linked List: 1 2 2 6 8

Union Linked List: 1 2 3 4 6 8 9

Input 2 :

6

1 5 1 2 2 5

5

4 5 6 7 1

Output 2 :

First Linked List: 1 1 2 2 5 5

Second Linked List: 1 4 5 6 7

Union Linked List: 1 2 4 5 6 7

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

struct Node{
   int data;
   Node *next;

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

```
void insertSorted(Node*&head, int val){
    Node* newNode = new Node(val);

    if(head == NULL || val < head->data){
        newNode->next = head;
        head = newNode;
        return;
    }

    Node* temp = head;
    while(temp->next != NULL && temp->next->data < val){
        temp = temp->next;
    }

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

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

Node* unionList(Node* l1, Node* l2){
    Node* result = nullptr;
    set<int> seen;
```

```
    while(l1){
        seen.insert(l1->data);
        l1 = l1->next;
    }

    while(l2){
        seen.insert(l2->data);
        l2 = l2->next;
    }

    for(int val : seen){
        insertSorted(result, val);
    }

    return result;
}

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

    int num;
    Node* l1 = nullptr;

    for(int i=0; i<n; i++){
        cin>>num;
        insertSorted(l1, num);
    }

    int m;
    cin>>m;
```

```
    Node* l2 = nullptr;

    for(int i=0; i<m; i++){
        cin>>num;
        insertSorted(l2, num);
    }

    cout<<"First Linked List: ";
    printLL(l1);
    cout<<endl;

    cout<<"Second Linked List: ";
    printLL(l2);
    cout<<endl;

    Node* uni = unionList(l1, l2);
    cout<<"Union Linked List: ";
    printLL(uni);



    return 0;
}
```

Problem Statement 7

You are tasked with creating a program that processes a singly linked list containing integer data and rearranges its nodes. Specifically, your program should separate the even and odd integers, placing the even integers before the odd ones while maintaining their original order within each group. Insert the new nodes at the beginning of the list.

Example

Input:

linked list = 1 2 3 4

Output:

4 2 3 1

Explanation:

Initially, the linked list contains four nodes with the values: 1 2 3 4.

The new nodes are inserted at the beginning of the list. So, after insertion, the nodes will be in the order: 4 3 2 1.

Now, rearrange the linked list to position even numbers before odd numbers while preserving their order.

The resulting rearranged linked list is 4 2 3 1.

Input format :

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

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

Output format :

The output displays the rearranged list, containing the even numbers followed by the odd numbers.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq N \leq 15$

$1 \leq$ elements ≤ 150

Sample test cases :

Input 1 :

4

1 2 3 4

Output 1 :

4 2 3 1

Input 2 :

5

12 15 13 14 16

Output 2 :

16 14 12 13 15

```
// 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){
        if(temp->data%2==0){
            cout<<temp->data<<" ";
        }
        temp = temp->next;
    }

    temp = head;
```

```
    while(temp != NULL){
        if(temp->data%2 != 0){
            cout<<temp->data<<" ";
        }
        temp = temp->next;
    }
}

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

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

    printLL(head);

    return 0;
}
```

Problem Statement 8

Sara has a series of sales data recorded in a singly linked list. She wants to left-shift the list by k nodes to better analyze the sales patterns.

Help Sara implement a program to left-shift her sales data linked list by k nodes.

Input format :

The first line of input contains an integer **N,** representing the number of nodes.

The second line consists of **N** space-separated integers, representing the sales data values.

The last integer **k** represents the number of nodes by which the list should be left-shifted.

Output format :

The output prints the updated linked list after left-shifting by k nodes.

Refer to the sample output for formatting specifications.

Code constraints :

1 ≤ N ≤ 25

1 ≤ data values ≤ 150

Sample test cases :

Input 1 :

5

2 4 7 8 9

3

Output 1 :

8 9 2 4 7

Input 2 :

8

1 2 3 4 5 6 7 8

4

Output 2 :

5 6 7 8 1 2 3 4

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

struct Node {
   int data;
```

```
    Node* next;
};

Node* createNode(int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

Node* insertNode(Node* head, int data) {
    if (head == NULL)
        head = createNode(data);
    else {
        Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = createNode(data);
    }
    return head;
}

Node* leftShiftLinkedList(Node* head, int k) {
    if(head == NULL || k == 0) return head;

    int len =0;
    Node* temp = head;
    while(temp != NULL){
        len++;
        temp = temp->next;
    }
```

```
    k = k%len;
    if( k ==0) return head;

    Node* curr = head;
    int count =1;
    while(curr != NULL && count < k){
        curr = curr->next;
        count++;
    }

    if(curr == NULL || curr->next == NULL) return head;

    Node* newHead = curr->next;
    curr->next = NULL;

    Node* tail = newHead;

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

    tail->next = head;

    return newHead;
}

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

```
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    int N, k;
    cin >> N;

    Node* head = NULL;
    for (int i = 0; i < N; i++) {
        int value;
        cin >> value;
        head = insertNode(head, value);
    }

    cin >> k;

    head = leftShiftLinkedList(head, k);

    printList(head);

    return 0;
}
```

Problem Statement 9

Aaron is building a text editor feature to check if a user-entered word is a palindrome using a singly linked list. Each character of the word is stored as a node in the linked list. Write a program for him to construct the linked list from user input and determine if the word is a palindrome.

Company Tags: Amazon, Microsoft, Snapdeal

Input format :

The input consists of a series of space-separated characters (alphabets, digits, special symbols). Enter $ to stop.

Output format :

The output prints whether the given characters form a palindrome or not.

Refer to the sample output for the exact text and format.

Code constraints :

The input will be valid ASCII characters.

Sample test cases :

Input 1 :

1 2 3 4 5 4 3 2 1 $

Output 1 :

The linked list is a palindrome

Input 2 :

a b c d e f $

Output 2 :

The linked list is not a palindrome

Input 3 :

! @ # % ^ % # @ ! $

Output 3 :

The linked list is a palindrome

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

struct Node {
   char data;
   Node* next;
```

```
    Node(char 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;
    }
}

bool isPalindrome(Node* head){
    vector<char> chars;

    Node* temp = head;
    while(temp != nullptr){
        chars.push_back(temp->data);
        temp = temp->next;
    }

    int start =0;
    int end = chars.size() -1;
```

```
    while(start < end) {
        if(chars[start] != chars[end]){
            return false;
        }
        start++;
        end--;
    }

    return true;
}

int main() {
    Node* head = nullptr;
    char ch;

    while(cin>> ch && ch != '$') {
        insertAtEnd(head, ch);
    }

    if(isPalindrome(head)) {
        cout<<"The linked list is a palindrome"<<endl;
    }else{
        cout<<"The linked list is not a palindrome"<<endl;
    }

    return 0;
}
```

Problem Statement 10

Saran is taked with writing a program that reads a sequence of characters to create a singly linked list, then prompts for an index and a character to insert after that index. Finally, he wants to display the updated linked list.

Help Saran in completing this program.

Input format :

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

The second line contains a sequence of **n** characters.

The third line contains an integer representing the index(0- based) after which the new character node needs to be inserted.

The fourth line contains a character value representing the character to be inserted after the given index.

Output format :

The output prints the updated linked list after the insertion of the new character node, after the given index.

If the provided index is out of bounds (larger than the list size), the program should output "Invalid index." and the linked list must remain unchanged.

Refer to the sample output for formatting specifications.

Code constraints :

The character value can be any printable ASCII character.

Sample test cases :

Input 1 :

5

a b c d e

2

X

Output 1 :

Updated list: a b c X d e

Input 2 :

3

x y z

0

A

Output 2 :

Updated list: x A y z

Input 3 :

4

a b c d

6

x

Output 3 :

Invalid index.

Updated list: a b c d

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

struct Node{
    char data;
    Node* next;

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

void insertAtEnd(Node*&head, char ch){
    Node* newNode = new Node(ch);

    if(head == nullptr){
```

```
        head = newNode;
    }else{
        Node* temp = head;
        while(temp->next != nullptr){
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

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

    if(index == 0){
        newNode->next = head->next;
        head->next = newNode;
        return;
    }


    Node* temp = head;
    for(int i=0; i<index  && 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;

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

    int index;
    cin>>index;
    char newchar;
    cin>>newchar;

    if(index >= n){
        cout<<"Invalid index."<<endl;
    }

    insertAtPos(head, index, newchar);

    cout<<"Updated list: ";
```

```
    printLL(head);

    return 0;
}
```

Problem Statement 11

A software engineer is developing a function for a new feature in an editor application that rearranges numerical data. To test this, they need to create a singly linked list where each node contains an integer.

The task is to swap every two adjacent nodes in the linked list and print the list before and after swapping.

Company Tags: Amazon, Microsoft, Moonfrog Labs

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** integers, representing the data values for each node in the linked list.

Output format :

The first line of output prints the linked list before swapping.

The second line prints the linked list after swapping.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq n \leq 30$

$0 \leq$ data values ≤ 100

Sample test cases :

Input 1 :

6

3 1 5 4 2 8

Output 1 :

Before swapping: 3 1 5 4 2 8

After swapping: 1 3 4 5 8 2

Input 2 :

5

1 2 3 4 5

Output 2 :

Before swapping: 1 2 3 4 5

After swapping: 2 1 4 3 5

```
// 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;
    }
}

Node* swapPairs(Node* head){
    if(!head || !head->next) return head;

    Node* dummy = new Node(0);
    dummy->next = head;
    Node* prev = dummy;

    while(prev->next && prev->next->next){
        Node* first = prev->next;
        Node* second = first->next;

        first->next = second->next;
        second->next = first;
        prev->next = second;

        prev = first;
    }
    return dummy->next;
}

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

```
    }
}

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

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

    cout<<"Before swapping: ";
    printLL(head);
    cout<<endl;

    head = swapPairs(head);

    cout<<"After swapping: ";
    printLL(head);

    return 0;
}
```

Problem Statement 12

A company wants to analyze the performance of its sales team by reviewing the sales figures from the most recent months. Each month's sales data is entered into a singly linked list, with each node representing a month's sales.

Write a program that computes the sum of sales for the last m months from this linked list data.

Input format :

The first line consists of an integer **n,** representing the number of months' sales data.

The second line consists of **n** space-separated integers, representing the sales data for each month.

The third line consists of an integer **m,** representing the number of most recent months to sum up.

Output format :

The output prints the sum of sales for the last **m** months.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq n \leq 100$

$0 \leq$ sales data ≤ 100

Sample test cases :

Input 1 :

6

3 1 0 4 30 12

3

Output 1 :

46

Input 2 :

10

7 4 1 2 5 8 0 3 6 9

5

Output 2 :

26

```
// 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;
}

int sumOfLastN(Node* head, int n){
    int sum =0;
    Node* temp = head;
    int i=0;
    while(i<n && temp != nullptr){
        sum += temp->data;
        temp = temp->next;
        i++;
    }
    return sum;
}

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

```
    Node* head = nullptr;

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

    int m;
    cin>>m;

    cout<<sumOfLastN(head, m);

    return 0;
}
```