Problem Statement 1

John is learning about grounded header linked lists in his computer science class. He wants to create a linked list where each node contains a character and then display the list. Implement a program that allows John to input characters at the end into a grounded header linked list and then display the contents of the list.

Company Tags: Capgemini

Input format :

The first line contains an integer **n,** the number of elements to be inserted in the linked list.

The next **n** lines each contain a single character, which will be inserted into the linked list.

Output format :

The output should be a single line containing the characters in the linked list, separated by a space.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 20$

The input will be lowercase and uppercase characters.

Sample test cases :

Input 1 :

5

a

b

c

d

e

Output 1 :

a b c d e

Input 2 :

3

A

B

C

Output 2 :

A B C

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

struct Node{
char data;
Node* next;

Node(char val = '\0'){
 data = val;
 next = nullptr;
}
};

void insertAtEnd(Node*&header, char ch){
 Node* newNode = new Node(ch);
 Node* temp = header;
 while(temp->next != nullptr){
  temp = temp->next;
 }
 temp->next = newNode;
}

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

```
  cout<<endl;
}

int main(){
  int n;
  cin>>n;
  Node* header = new Node();

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

  printLL(header);

  return 0;
}
```

Problem Statement 2

In a bustling office, Sarah needs to keep track of her tasks for the day. She wants to keep adding new tasks at the beginning of her to-do list.

Help Sarah by writing a program that maintains a grounded header linked list of tasks where each new task is added at the beginning of the list.

Input format :

The input consists of a series of integers, each representing a task. The input ends when -1 is entered, which indicates the end of the input.

Output format :

The output prints the grounded header linked list where each integer represents a task, separated by a space. The most recently added task should appear first.

Refer to the sample output for formatting specifications.

Code constraints :

1 ≤ tasks ≤ 100

Sample test cases :

Input 1 :

15

23

36

48

-1

Output 1 :

48 36 23 15

Input 2 :

78

85

91

17

65

-1

Output 2 :

65 17 91 85 78

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

struct Node{
int data;
Node* next;

Node(int val = 0){
  data = val;
```

```
  next = nullptr;
}
};

void insertAtFront(Node*& header, int num){
  Node* newNode = new Node(num);
  newNode->next = header->next;
  header->next = newNode;
  return;
}

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

int main(){
  Node* header = new Node();

  int num;

  while(true){
    cin>>num;
    if(num == -1){
      break;
    }
    insertAtFront(header, num);
```

```
 }

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

Problem Statement 3

Madhan is developing a contact management system for a mobile application. The system allows users to maintain a list of their contacts. Whenever a user adds a new contact, the system should add it at the end of the contact list.

Write a program for Madhan to implement the code to perform insertion at the end using a grounded header linked list.

Company Tags: TCS

Input format :

The first line of input consists of the size **n** of the contact list.

The second line consists of **n** contact elements.

Output format :

The output represents the contact list with the contacts added at the end.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq n \leq 20$

$1 \leq$ contact elements ≤ 100

Sample test cases :

Input 1 :

5

1 2 3 4 5

Output 1 :

1 2 3 4 5

Input 2 :

8

13 45 67 82 29 35 72 91

Output 2 :

13 45 67 82 29 35 72 91

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

struct Node{
int data;
Node* next;

Node(int val = 0){
 data = val;
 next = nullptr;
}
};

void insertAtEnd(Node*&header, int num){
 Node* newNode = new Node(num);
 Node* temp = header;
 while(temp->next != nullptr){
  temp = temp->next;
 }
 temp->next = newNode;
}

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

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

int main(){
  int n;
  cin>>n;
  Node* header = new Node();

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

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

Problem Statement 4

Suppose you are working on a text editing application and you need to implement a feature that allows users to insert a character at a specific index in the text. You decide to implement this feature using a grounded header linked list to manage the text efficiently.

Company Tags: Capgemini

Input format :

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

The second line consists of **n** space-separated characters.

The third line consists of an integer **index** representing the position(0- based) for insertion.

The fourth line consists of a character to be inserted at the specified index.

Output format :

The output prints "Updated list: " followed by the list elements after inserting the character at the specified index.

Print "Invalid position." for invalid inputs.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$0 \leq n \leq 10$

Sample test cases :

Input 1 :

5

A B C D E

2

X

Output 1 :

Updated list: A B X C D E

Input 2 :

3

P Q R

0

S

Output 2 :

Updated list: S P Q R

Input 3 :

7

p q r s t u v

10

l

Output 3 :

Invalid position.

Updated list: p q r s t u v

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

struct Node{
int data;
Node* next;

Node(int val=0){
 data = val;
 next = nullptr;
}
};

void insertAtEnd(Node*&header, int val){
 Node* newNode = new Node(val);

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

void insertAtPosition(Node*& header, int pos, int val){
 Node* newNode = new Node(val);
 if(pos == 1){
  newNode->next = header->next;
  header->next = newNode;
  return;
```

```
  }

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

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

}

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



int main(){
  int n;
  cin>>n;
  Node* header = new Node();

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

```
  insertAtEnd(header, num);
 }

 int m;
 cin>>m;
 int val;
 cin>>val;

 insertAtPosition(header, m, val);

 cout<<"Updated list: ";
 printLL(header);

 return 0;
}
```

Problem Statement 5

Alex is maintaining a list of integers representing stock quantities in his warehouse. Sometimes, he needs to remove all occurrences of a particular quantity from the list when an item is discontinued. He wants to use a grounded header linked list for this purpose.

Help Alex in the task.

Company Tags: CTS

Input format :

The first line of input consists of the number of items **n** in the warehouse.

The second line consists of **n** elements, separated by space.

The third line consists of the quantity to remove all occurrences.

Output format :

The output prints the sequence of integers representing the updated list of stock quantities after the removal of all occurrences of the specified value.

Refer to the sample output for formatting specifications.

Code constraints :

1 ≤ n ≤ 15

1 ≤ elements ≤ 100

Sample test cases :

Input 1 :

5

1 2 3 3 3

3

Output 1 :

1 2

Input 2 :

6

1 1 1 1 1 12

1

Output 2 :

12

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

struct Node{
int data;
Node* next;

Node(int val = 0){
 data = val;
 next = nullptr;
}
};

void insertAtEnd(Node*& header, int val){
```

```
  Node* newNode = new Node(val);

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

void removeAllOccurrences(Node*&header, int key){
  Node* prev = header;
  Node* curr = header->next;

  while(curr != nullptr){
    if(curr->data == key){
      prev->next = curr->next;
      delete curr;
      curr = prev->next;
    }else{
      prev = curr;
      curr = curr->next;
    }
  }
}

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

```
}

int main(){
 int n;
 cin>>n;
 Node* header = new Node();

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

 int key;
 cin>>key;

 removeAllOccurrences(header, key);

 printLL(header);

 return 0;

}
```

Problem Statement 6

In a warehouse, a system keeps track of packages using a grounded header linked list where each node represents a package. Each package is added to the end of the list. Sometimes, packages need to be removed from specific positions in the list.

Your task is to implement this system which can add packages to the list, display the list, and remove a package from a specified position.

Input format :

The first line contains an integer, **n,** which represents the number of packages to be added to the list.

The second line contains **n** integers, each representing the ID of a package.

The third line contains an integer, **position,** which indicates the position of the package to be removed from the list (1-based index).

Output format :

The first line of output represents the elements before deletion and the next line represents the remaining elements in the linked list after deleting the particular value.

Refer to the sample output for formatting specifications.

Code constraints :

1 ≤ n ≤ 25

1 ≤ elements ≤ 100

Sample test cases :

Input 1 :

5

10 20 30 40 50

2

Output 1 :

Linked List before deletion: 10 20 30 40 50

Linked List after deletion: 10 30 40 50

Input 2 :

5

12 23 43 56 89

5

Output 2 :

Linked List before deletion: 12 23 43 56 89

Linked List after deletion: 12 23 43 56

```
#include<iostream>

using namespace std;

struct Node{
```

```
int data;
Node* next;

Node(int val=0){
  data = val;
  next = nullptr;
}
};

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

void deleteAtIndex(Node*&header, int index){
  Node* prev = header;
  Node* temp = header->next;

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

  if(temp == nullptr) return;

  prev->next = temp->next;
  delete temp;
```

```
}

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

int main(){
  int n;
  cin>>n;
  Node* header = new Node();

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

  int pos;
  cin>>pos;

  deleteAtIndex(header, pos);

  printLL(header);

  return 0;
}
```

Problem Statement 7

Suppose you are working on a student management system for a university, and you need to implement a deletion functionality for a grounded header linked list to manage student records. Write a program to implement deletion logic with multiple deletions based on user input.

Company Tags: TCS

Input format :

The first line of input consists of the elements to be inserted in the list.

The second line consists of the elements to be deleted.

Terminate the inputs when -1 is given.

Output format :

The output prints the remaining elements in the reverse order, after deleting the given element.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

1 ≤ list elements ≤ 100

Sample test cases :

Input 1 :

1 2 3 4 5 -1

2 -1

Output 1 :

5 4 3 1

Input 2 :

1 2 3 4 5 -1

2 3 4 -1

Output 2 :

5 1

```
#include<iostream>
```

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

struct Node{
int data;
Node* next;

Node(int val=0){
  data = val;
  next = nullptr;
}
};

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

void deleteNode(Node*&header, int val){
  Node* prev = header;
  Node* curr = header->next;

  while(curr != nullptr){
    if(curr->data == val){
      prev->next = curr->next;
      delete curr;
      curr = prev->next;
```

```
    }else{
      prev = curr;
      curr = curr->next;
    }
  }
}

void printReverse(Node* header){
  vector<int> vals;
  Node* temp = header->next;

  while(temp != nullptr){
    vals.push_back(temp->data);
    temp = temp->next;
  }

  for(int i = vals.size() -1; i >=0; i--){
    cout<<vals[i]<<" ";
  }
  cout<<endl;
}


int main(){
  Node* header = new Node();

  int num;
  while(cin>>num && num != -1){
    insertAtEnd(header, num);
  }
```

```
 while(cin>>num && num != -1){
   deleteNode(header, num);
 }

 printReverse(header);

 return 0;
}
```

Problem Statement 8

Emma is working with two sets of sensor data recorded in two separate grounded header linked lists. She wants to merge these two lists alternately to analyze the combined data more efficiently.

Help Emma by writing a program to merge the two grounded header linked lists alternately.

Company Tags: TCS

Input format :

The first line of input consists of the number of elements **n** in the first list.

The second line consists of **n** elements, separated by space.

The third line consists of the number of elements **m** in the second list.

The fourth line consists of **m** elements, separated by space.

Output format :

The output prints the merged linked list with nodes from both lists alternately.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n, m \leq 15$

$1 \leq$ elements ≤ 100

Sample test cases :

Input 1 :

5

1 2 3 4 5

5

6 7 8 9 10

Output 1 :

1 6 2 7 3 8 4 9 5 10

Input 2 :

4

12 26 34 48

4

56 59 64 78

Output 2 :

12 56 26 59 34 64 48 78

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

struct Node{
int data;
Node* next;

Node(int val=0){
 data = val;
 next = nullptr;
}
};

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

 Node* temp = header;
 while(temp->next != nullptr){
```

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

Node* mergeAlternateNode(Node* header1, Node* header2){
  Node* temp1 = header1->next;
  Node* temp2 = header2->next;

  Node* mergedHeader = new Node();
  Node* mergedTail = mergedHeader;

  while(temp1 != nullptr && temp2 != nullptr){
    mergedTail->next = new Node(temp1->data);
    mergedTail = mergedTail->next;
    temp1 = temp1->next;

    mergedTail->next = new Node(temp2->data);
    mergedTail = mergedTail->next;
    temp2 = temp2->next;
  }

  while(temp1 != nullptr){
    mergedTail->next = new Node(temp1->data);
    mergedTail = mergedTail->next;
    temp1 = temp1->next;
  }

  while(temp2 != nullptr){
    mergedTail->next = new Node(temp2->data);
    mergedTail = mergedTail->next;
```

```
    temp2 = temp2->next;
  }
  return mergedHeader;
}

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

int main(){
  int n;
  cin>>n;
  Node* header1 = new Node();
  Node* header2 = new Node();

  for(int i=0; i<n; i++){
    int num;
    cin>>num;
    insertAtEnd(header1, num);
  }

  int m;
  cin>>m;

  for(int i=0; i<m; i++){
    int num;
    cin>>num;
```

```
    insertAtEnd(header2, num);
  }

  Node* mergedHeader = mergeAlternateNode(header1, header2);

  printLL(mergedHeader);

  return 0;
}
```

Problem Statement 9

Sharon is managing a list of items in her online store. She needs to track and display the number of items added to her inventory each day. To do this, she decides to use a grounded header linked list structure where each node represents an item added to the inventory.

Guide Sharon in completing the program.

Input format :

The input consists of a sequence of integer values representing item quantities, followed by -1 which signals the end of input.

Output format :

The output prints an integer representing the number of items in the inventory.

Refer to the sample output for formatting specifications.

Code constraints :

1 ≤ item quantity ≤ 100

Sample test cases :

Input 1 :

1 2 3 4 5 -1

Output 1 :

5

Input 2 :

12 34 78 92 46 78 -1

Output 2 :

6

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

struct Node{
int data;
Node* next;

Node(int val =0){
 data = val;
 next = nullptr;
}
};

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

int countNode(Node* header){
 int count =0;
 Node* temp = header->next;
 while(temp != nullptr){
  count++;
  temp = temp->next;
 }
```

```
 return count;
}

int main(){
 Node* header = new Node();

 int x;
 while(true){
  cin>>x;
  if(x == -1){
   break;
  }
  insertAtEnd(header, x);
 }

 cout<<countNode(header);

 return 0;
}
```

Problem Statement 10

In a small town, Alice often rearranges the list of tasks she needs to complete. She wants to rotate the list of tasks to the right by a given number of positions.

Help Alice by implementing a program that performs this rotation efficiently using a grounded header linked list.

Company Tags: Capgemini

Input format :

The first line of input consists of the number of elements **n** in the task list.

The second line consists of **n** elements, separated by space.

The third line consists of the number of positions to rotate right in the task list.

Output format :

The first line of output prints the original list.

The second line prints the rotated list.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 20$

$1 \leq$ list elements ≤ 100

Sample test cases :

Input 1 :

5

1 2 3 4 5

3

Output 1 :

Original List: 1 2 3 4 5

Rotated List: 3 4 5 1 2

Input 2 :

6

25 36 95 74 86 12

1

Output 2 :

Original List: 25 36 95 74 86 12

Rotated List: 12 25 36 95 74 86

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

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

```
Node(int val =0){
  data = val;
  next = nullptr;
}
};

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

int countNode(Node* header){
  int count =0;
  Node* temp = header->next;
  while(temp != nullptr){
    count++;
    temp = temp->next;
  }
  return count;
}

void rotateRight(Node*& header, int k){
  int n = countNode(header);
  if(n==0 ||n%k == 0) return;

  k = k%n;
```

```
  int leftShift = n - k;

  Node* prev = header;
  for(int i=0; i<leftShift; i++){
    prev = prev->next;
  }

  Node* newHead = prev->next;
  Node* tail = newHead;

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

  tail->next = header->next;
  header->next = newHead;
  prev->next = nullptr;
}

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

int main(){
  Node* header = new Node();
```

```
 int n;
 cin>>n;

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

 int k;
 cin>>k;

 cout<<"Original list: ";
 printLL(header);
 rotateRight(header, k);
 cout<<"Rotated list: ";
 printLL(header);

 return 0;
}
```

Problem Statement 11

In a university, students are registering for different clubs based on their student ID numbers. Each club wants to separate students into two lists: one for students with even ID numbers and another for students with odd ID numbers.

Your task is to write a program to help the clubs split the list of student IDs into two separate lists based on their parity (even or odd), using a grounded header linked list.

Company Tags: Infosys

Input format :

The first line contains an integer, **n,** representing the number of student IDs.

The second line contains **n** integers, each representing a student ID.

Output format :

The first line of output prints the list of student IDs with even numbers.

The second line prints the list of student IDs with odd numbers.

Refer to the sample output for formatting specifications.

Code constraints :

$1 \leq n \leq 20$

$1 \leq$ student IDs ≤ 100

Sample test cases :

Input 1 :

5

14 26 38 45 51

Output 1 :

Even List: 14 26 38

Odd List: 45 51

Input 2 :

8

41 52 74 46 78 90 42 94

Output 2 :

Even List: 52 74 46 78 90 42 94

Odd List: 41

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

struct Node{
int data;
Node* next;

Node(int val=0){
```

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

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

void printEven(Node* header){
  Node* temp = header->next;
  while(temp != nullptr){
    if(temp->data%2 == 0){
      cout<<temp->data<<" ";
    }
    temp = temp->next;
  }
}

void printOdd(Node* header){
  Node* temp = header->next;
  while(temp != nullptr){
    if(temp->data%2 != 0){
      cout<<temp->data<<" ";
    }
    temp = temp->next;
```

```
  }
}

int main(){
  int n;
  cin>>n;
  Node* header = new Node();

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

  cout<<"Even list: ";
  printEven(header);
  cout<<endl;
  cout<<"Odd list: ";
  printOdd(header);

  return 0;
}
```

Problem Statement 12

Racheal is tasked with implementing a student grades matrix using the sparse matrix representation with a grounded header linked list. The matrix will store the grades of students for different subjects.

Write logic for Racheal to implement a sparse matrix using a grounded header linked list.

Company Tags: Infosys

Input format :

The first line represents a row **r** of the matrix.

The second line represents column **c** of the matrix.

The remaining lines consist of the **r*c** matrix elements.

Output format :

The output represents the sparse matrix.

Refer to the sample output for formatting specifications.

Code constraints :

The given test cases fall under the following constraints:

$1 \leq r, c \leq 10$

Sample test cases :

Input 1 :

3

3

1 0 0

0 2 0

0 0 3

Output 1 :

1 0 0

0 2 0

0 0 3

Input 2 :

3

4

1 0 2 1

2 0 1 3

3 2 1 4

Output 2 :

1 0 2 1

2 0 1 3

3 2 1 4

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

struct Node{
int row, col, data;
Node* next;

Node(int r=0, int c=0, int val=0){
 data = val;
 row = r;
 col = c;
 next = nullptr;
}
};

void insert(Node*&header, int r, int c, int val){
 if(val == 0) return;

 Node* newNode = new Node(r,c,val);
 Node* temp = header;
 while(temp->next != nullptr){
  temp = temp->next;
 }
 temp->next = newNode;
}

void displayMatrix(Node* header, int r, int c){
 Node* temp = header->next;
 for(int i=0; i<r; i++){
  for(int j=0; j<c; j++){
    if(temp != nullptr && temp->row == i && temp->col == j){
```

```
      cout<<temp->data<<" ";
      temp = temp->next;
     }else{
      cout<<"0 ";
     }
   }
   cout<<endl;
  }
}

int main(){
  int r,c;
  cin>>r>>c;

  Node* header = new Node();

  int val;
  for(int i=0; i<r; i++){
    for(int j=0; j<c; j++){
      cin>>val;
      insert(header, i, j, val);
    }
  }

  displayMatrix(header, r,c);

  return 0;
}
```