Birla Institute of Technology and Science, Pilani
Hyderabad Campus

CS F211 Data Structures and Algorithms
Lab 2: Singly Linked Lists and Structures

Allowed languages: **C**

General Tips

- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, use comments wherever necessary.
- Use a proper IDE or text editors like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily. However in the lab you will be using command line interface.
- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.

Directions for Problems A-H

- Problems A-H are on singly linked lists. Do not use any other data structures (such as arrays) apart from linked lists to solve them.
- Assume that the elements of the singly linked lists can be both positive and negative unless mentioned otherwise.
- Use the following template to read input and print output:

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

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

// Creates a new node with given value and returns a pointer to it
Node *createNode(int value) {
    Node *newNode = malloc(sizeof(Node));
    assert(newNode != NULL);
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

// Creates a new node with given value and adds it to
// the back of the given singly linked list,
// returns a pointer to the newly created node
Node *addToList(Node *head, int val) {
    Node *newNode = createNode(val);
    if (head == NULL) {
        return newNode;
    }
    Node *cur = head;
    while (cur->next != NULL) {
        cur = cur->next;
    }
    cur->next = newNode;
    return newNode;
}

// Creates a singly linked list by reading input and
// returns a pointer the head of the newly created linked list
Node *readList() {
    int n;
    scanf("%d", &n);

    Node *head = NULL;

    for (int i = 0; i < n; ++i) {
        int x;
        scanf("%d", &x);
```

```
        if (head == NULL) {
            head = addToList(head, x);
        } else {
            addToList(head, x);
        }
    }

    return head;
}

// Prints the values stored in the nodes of the given singly linked list
void printList(Node *head) {
    Node *ptr = head;
    while (ptr != NULL) {
        printf("%d ", ptr->data);
        ptr = ptr->next;
    }
    printf("\n");
    return;
}

// Frees the memory dynamically allocated to
// all the nodes of the given singly linked list
void freeList(Node *head) {
    Node *cur, *nxt;
    cur = head;
    while (cur != NULL) {
        nxt = cur->next;
        free(cur);
        cur = nxt;
    }
}

// Function definition goes here



int main(void) {

    // Function calls go here


    return 0;
}
```

- Do not modify any of the functions present in the template. Every linked list problem includes a function prototype and some statements to be added to `main`. Copy them to the template

and complete the function definition to solve the problem.

- Do not modify the function prototype. You can add helper functions if required.
- **The code template for singly linked list problems will not be provided to you during lab evaluation. For lab evaluation, you need to code everything from scratch.**
- Do not add global variables.
- If a problem requires you to return a list, then you must return a pointer to the head of the required list.
- Check your program for memory leaks.

Problem A. Find Median

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

Given a sorted singly linked list, find the median of the list. Median for a list of size n is defined as follows:

- If n is odd, then the median is the $(\frac{n+1}{2})$-th element of the list.
- If n is even, then the median is the average of the $(\frac{n}{2})$-th and $(\frac{n+1}{2})$-th element, rounded down.

Assume 1-based indexing.

Input

The first line of input contains n — the number of elements in the singly linked list.

The second line of input contains n space separated integers — the elements of the singly linked list.

Output

Print one line containing the median of the singly linked list.

Examples

standard input	standard output
5 1 2 3 5 7	3
6 -5 -2 5 8 8 9	6

Note

Use the following function prototype:

```
int findMedian(Node *head)
```

Add the following to `main`:

```
Node *head = readList();
int median = findMedian(head);
printf("%d\n", median);
freeList(head);
```

Problem B. Merge Lists

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

Rahul was given a singly linked list to sort. He broke the list into two smaller parts and was able to sort them. He now wants your help in merging them. Given the sorted lists from Rahul, merge them into a single sorted list and return the merged list.

Input

The first line of input contains n — the number of elements in the first singly linked list.

The second line of input contains n space separated integers — the elements of the first singly linked list.

The third line of input contains m — the number of elements in the second singly linked list.

The fourth line of input contains m space separated integers — the elements of the second singly linked list.

Output

Print one line containing the elements of the merged list, separated by space.

Examples

standard input	standard output
4 3 4 6 7 4 1 5 6 8	1 3 4 5 6 6 7 8
3 -5 0 11 5 -1 2 2 3 4	-5 -1 0 2 2 3 4 11

Note

Use the following function prototype:

```
Node *mergeLists(Node *head1, Node *head2)
```

Add the following to `main`:

```
Node *head1 = readList();
Node *head2 = readList();
Node *merged = mergeLists(head1, head2);
printList(merged);
freeList(head1);
freeList(head2);
freeList(merged);
```

Problem C. Sort List

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

Alice likes sorted lists as they make it easier for her to work with them. Alice is given a singly linked list to perform some task and she asks for your help to sort the list so that she can finish the task faster. Given the list from Alice, sort it and return the sorted list.

If you simply print the list in sorted order instead of actually sorting the list, full marks will not be awarded even for correct output. Also, sorting a list implies modifying the node pointers to create a sorted list, not simply exchanging node elements.

Hint: You can use your solutions for Problem A and Problem B to find an optimal solution. However, it is not mandatory to find an optimal solution at this point.

Input

The first line of input contains n — the number of elements in the singly linked list.

The second line of input contains n space separated integers — the elements of the singly linked list.

Output

Print one line containing the elements of the sorted list, separated by space.

Example

standard input	standard output
6 4 5 1 -6 3 2	-6 1 2 3 4 5

Note

Use the following function prototype:

```
Node *sortList(Node *head)
```

Add the following to `main`:

```
Node *head = readList();
Node *sorted = sortList(head);
printList(sorted);
freeList(head);
freeList(sorted);
```

Problem D. Reverse List

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

Messi has been learning Arabic, but he keeps writing integer sequences in reverse. As his new best friend, you must correct his mistakes. Given a singly linked list containing integers written by Messi, reverse the list and return the resulting list. You are not allowed to simply print the elements of the linked list in reverse order.

Input

The first line of input contains n — the number of elements in the singly linked list.

The second line of input contains n space separated integers — the elements of the singly linked list.

Output

Print one line containing the elements of the reversed list, separated by space.

Example

standard input	standard output
6 5 4 7 8 12 6	6 12 8 7 4 5

Note

Use the following function prototype:

```
Node *reverseList(Node *head)
```

Add the following to `main`:

```
Node *head = readList();
head = reverseList(head);
printList(head);
freeList(head);
```

Problem E. Palindrome List

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

Given a list of integers in the form of a singly linked list, check if the list is a palindrome or not. A palindrome is a sequence that reads the same forward and backward. If the list is a palindrome, print 1, else print 0.

You are not allowed to create a new linked list.

Input

The first line of input contains n — the number of elements in the singly linked list.

The second line of input contains n space separated integers — the elements of the singly linked list.

Output

Print one line containing 1 if the list is a palindrome, 0 otherwise.

Examples

standard input	standard output
6 1 2 4 4 2 1	1
5 2 6 7 7 6	0

Note

Use the following function prototype:

```
int isPalindrome(Node *head)
```

Add the following to `main`:

```
Node *head = readList();
int palin = isPalindrome(head);
printf("%d\n", palin);
freeList(head);
```

Problem F. Paper Mixup

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

It's compre time and you are the invigilator in an examination hall containing CS and ECE students. The CS and ECE students are seated alternately. When it came to collecting the papers, you forgot to collect the papers of the CS and ECE students separately, resulting in alternating papers with papers for the ECE course being in the odd positions and the CS ones in the even positions. You now have to rearrange the papers such that -

- All the ECE papers are placed before all the CS papers
- The relative ordering of both the ECE and CS papers should be maintained i.e if the CS paper numbered 4 appears before the CS paper numbered 3, in the original arrangement, the CS paper numbered 4 should appear before the CS paper numbered 3 in the new arrangement as well (Check example for better understanding)

The list of papers is given as a singly linked list. Rearrange the singly linked list and find the final arrangement of the papers formed by following the above conditions. You are not allowed to create a new singly linked list.

Note: The first paper is considered to be in the odd position. Also, it is not guaranteed that the number of ECE papers and CS papers are equal.

Input

The first line of input contains n — the number of papers collected.

The next line contains n space separated integers — the elements of the singly linked list.

Output

Print the final arrangement of the linked list.

Examples

standard input	standard output
7 4 1 6 2 3 9 7	4 6 3 7 1 2 9
6 -1 4 2 -5 8 2	-1 2 8 4 -5 2

Note

Use the following function prototype:

```
Node* rearrangeList(Node *head)
```

Add the following to `main`:

```
Node *head = readList();
Node *newList = rearrangeList(head);
printList(newList);
freeList(head);
freeList(newList);
```

Problem G. CMS Error

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

The database of your college's CMS has become corrupted. Initially, it was in the form of a sorted singly linked list containing unique integers (i.e., no integer was repeated in the list). As a result of the corruption, some nodes got infected and created copies of themselves. The number of nodes that got corrupted may be zero or more. Given the corrupted linked list, **remove all nodes containing values that appear more than once** in the list and return the resultant list.

Note that the corrupted list is still sorted, and your resultant list must be sorted as well. You are not allowed to create a new linked list. You need to modify the given one only.

Input

The first line of input contains n — the number of elements in the corrupted singly linked list.

The second line of input contains n space separated integers — the elements of the corrupted singly linked list.

Output

Print one line containing the elements of the resultant list, separated by space.

Examples

standard input	standard output
9 1 3 3 3 4 6 6 7 8	1 4 7 8
8 2 4 5 16 23 34 45 51	2 4 5 16 23 34 45 51

Note

In the first example, 3 and 6 appear more than once, so we remove all occurrences of 3 and 6 to get the resultant list - 1, 4, 7, 8.

Use the following function prototype:

```
Node* removeRepeatedNodes(Node *head)
```

Add the following to `main`:

```
Node *head = readList();
Node *newList = removeRepeatedNodes(head);
printList(newList);
freeList(head);
freeList(newList);
```

Problem H. Extra Slide

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

Satya has prepared slides for his next presentation, but he forgot to delete a slide from the final presentation. The slides can be represented as a singly linked list of integers. Satya remembers that the slide that he had to delete was at the k-th position from the end of the list. Given the list and k, help Satya fix his slides by deleting the k-th slide from the end and returning the resulting singly linked list. It is guaranteed that the k-th slide from the end is present in the list.

Assume 1-based indexing from the end for the value of k.

Input

The first line of input contains two integers k, n — the position of the slide to be deleted from the end of the list and the number of elements of the singly linked list, respectively.

The second line of input contains n space separated integers — the elements of the singly linked list.

Output

Print one line containing the elements of the list after the deletion of the specified slide, separated by space.

Examples

standard input	standard output
4 5 7 21 45 63 10	7 45 63 10
1 5 1 4 2 9 3	1 4 2 9

Note

Use the following function prototype:

```
Node* removeSlide(Node *head, int k)
```

Add the following to `main`:

```
int k;
scanf("%d", &k);
Node *head = readList();
Node *corrected = removeSlide(head, k);
printList(corrected);
freeList(corrected);
freeList(head);
```

Problem I. BITS ID

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

You are tasked with generating the $BITS$ ID for a pair of $BITS$ Mail and Branch. Create a *structure* with 3 string members - the *email*, *branch name*, and the *id*. Create an instance of the structure to store the *email* and *branch name* from the input, then pass it by reference to a function to generate and store the *id*. It is guaranteed that the email will be of the form ***f20XXXXXX@hyderabad.bits-pilani.ac.in*** and the branch name will be one of CS, ECE, or EEE. Use the relevant portion of the email and the corresponding branch codes for the branches ($A7$ for CS, AA for ECE, $A3$ for EEE) to generate the *id*.

For the sake of simplicity, assume that all students are enrolled in PS and not TS.

Input

The first line of input contains a string − the $BITS$ Mail.

The second line contains another string − the $Branch\ name$.

Output

Print one line containing the $BITS\ ID$.

Examples

standard input	standard output
f20200010@hyderabad.bits-pilani.ac.in CS	2020A7PS0010H
f20212051@hyderabad.bits-pilani.ac.in ECE	2021AAPS2051H

Problem J. Nolympic Games

Input file:	`standard input`
Output file:	`standard output`
Time limit:	NA
Memory limit:	NA

The annual Nolympic Games concluded recently. n countries participated in it. You are given the names of each country along with their final medal tally, consisting of the number of *gold*, *silver* and *bronze* medals won by them. Create a structure to store the name of a country along with its medal tally. Subsequently, create an array of n elements, where each element is an instance of the structure and store the given input in the array. Finally, sort the array to determine the rankings of all the n countries.

The countries are first ranked according to the number of *gold* medals won by them (a country with higher *gold* medal count should have higher rank), if two countries have the same number of *gold* medals then ties are broken by their *silver* medal counts and finally by their *bronze* medal counts. It is guaranteed that no two countries will have the same number of *gold*, *silver* as well as *bronze* medals.

Input

The first line of input contains n — the number of countries.

The following n lines contain 4 elements each — the country name (a string), followed by 3 integers which are the *gold*, *silver* and *bronze* medal counts for that country respectively. It is guaranteed that the country name will not exceed 20 characters and will not contain any spaces.

Output

Print n lines − each containing the name of one country. The country names should be sorted from highest rank to lowest.

Example

standard input	standard output
4 China 4 2 10 USA 4 4 5 Qatar 0 5 2 India 1 4 1	USA China India Qatar