Rajalakshmi Engineering College

Name: Libin Mathew
Email: 241801137@rajalakshmi.edu.in
Roll no: 241801137
Phone: 9947119753
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 18

Section 1 : MCQ

1. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

void addFirst(int data){   Node* newNode = new Node(data);   newNode-&gt;next = head;   if (head != NULL) {   head-&gt;prev = newNode;   }   head = newNode;   }

Status : Correct ***Marks : 1/1***

2. What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
```

newNode->prev = NULL;

Answer

Creates a new node and initializes its data to 'value'

Status : Correct ***Marks : 1/1***

3. Consider the following function that refers to the head of a Doubly Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

```
Procedure fun(head_ref: Pointer to Pointer of node)
   temp = NULL
   current = *head_ref

   While current is not NULL
      temp = current->prev
      current->prev = current->next
      current->next = temp
      current = current->prev
   End While

   If temp is not NULL
      *head_ref = temp->prev
   End If
End Procedure
```

Answer

6 &lt;--&gt; 5 &lt;--&gt; 4 &lt;--&gt; 3 &lt;--&gt; 1 &lt;--&gt; 2.

Status : Wrong ***Marks : 0/1***

4. Which of the following is true about the last node in a doubly linked list?

Answer

Its next pointer is NULL

Status : Correct ***Marks : 1/1***

5. What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

Status : Correct ***Marks : 1/1***

6. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

Status : Correct ***Marks : 1/1***

7. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

Status : Correct ***Marks : 1/1***

8. Which of the following information is stored in a doubly-linked list's nodes?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

9. How many pointers does a node in a doubly linked list have?

Answer

2

Status : Correct *Marks : 1/1*

10. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));

Status : Correct *Marks : 1/1*

11. Which of the following is false about a doubly linked list?

Answer

Implementing a doubly linked list is easier than singly linked list

Status : Correct *Marks : 1/1*

12. Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

Status : Correct *Marks : 1/1*

13. Consider the provided pseudo code. How can you initialize an empty two-way linked list?

```
Define Structure Node
   data: Integer
   prev: Pointer to Node
   next: Pointer to Node
End Define
```

```
Define Structure TwoWayLinkedList
   head: Pointer to Node
   tail: Pointer to Node
End Define
```

Answer

struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list-&gt;head = NULL; list-&gt;tail = NULL;

Status : Correct ***Marks : 1/1***

14. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

15. What will be the output of the following program?

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

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

int main() {
   struct Node* head = NULL;
   struct Node* tail = NULL;
   for (int i = 0; i < 5; i++) {
      struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
      temp->data = i + 1;
      temp->prev = tail;
      temp->next = NULL;
      if (tail != NULL) {
            tail->next = temp;
        } else {
            head = temp;
        }
        tail = temp;
    }
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    return 0;
}
```

Answer

1 2 3 4 5

Status : Correct ***Marks : 1/1***

16. What is a memory-efficient double-linked list?

Answer

A doubly linked list that uses bitwise AND operator for storing addresses

Status : Correct ***Marks : 1/1***

17. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

Status : Correct ***Marks : 1/1***

18. What will be the output of the following code?

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

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = NULL;
    temp->prev = NULL;
    head = temp;
    printf("%d\n", head->data);
    free(temp);
    return 0;
}
```

Answer

2

Status : Correct ***Marks : 1/1***

19. Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {
    if (*head_ref == NULL || del_node == NULL) {
        return;
    }
    if (*head_ref == del_node) {
        *head_ref = del_node->next;
    }
    if (del_node->next != NULL) {
        del_node->next->prev = del_node->prev;
    }
    if (del_node->prev != NULL) {
        del_node->prev->next = del_node->next;
    }
    free(del_node);
}
```

Answer

Deletes the node at a given position in a doubly linked list.

Status : Wrong ***Marks : 0/1***

20. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {
      int Value;
      struct Node *Fwd;
      struct Node *Bwd;
);
```

Answer

X-&gt;Bwd-&gt;Fwd = X-&gt;Fwd; X-&gt;Fwd-&gt;Bwd = X-&gt;Bwd;

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Libin Mathew
Email: 241801137@rajalakshmi.edu.in
Roll no: 241801137
Phone: 9947119753
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bob is tasked with developing a company's employee record management system. The system needs to maintain a list of employee records using a doubly linked list. Each employee is represented by a unique integer ID.

Help Bob to complete a program that adds employee records at the front, traverses the list, and prints the same for each addition of employees to the list.

Input Format

The first line of input consists of an integer N, representing the number of employees.

The second line consists of N space-separated integers, representing the employee IDs.

Output Format

For each employee ID, the program prints "Node Inserted" followed by the current state of the doubly linked list in the next line, with the data values of each node separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
101 102 103 104

Output: Node Inserted
101
Node Inserted
102 101
Node Inserted
103 102 101
Node Inserted
104 103 102 101

Answer

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

struct node {
    int info;
    struct node* prev, * next;
};

struct node* start = NULL;

struct node* head=NULL;// You are using GCC
void traverse() {
   struct node* temp=head;
   while(temp!=NULL)
   {
      printf("%d",temp->info);
      temp=temp->next;
   }
   printf("\n");
}

void insertAtFront(int data) {
   struct node* newNode=(struct node*)malloc(sizeof(struct node));
   newNode->info=data;
   newNode->next=NULL;
   newNode->prev=NULL;
   if(head==NULL)
   {
      head=newNode;
      printf("Node Inserted\n");
      return;
   }
   newNode->next=head;
   head->prev=newNode;
   printf("Node Inserted\n");
   head=newNode;
}


int main() {
    int n, data;
    cin >> n;
    for (int i = 0; i < n; ++i) {
       cin >> data;
       insertAtFront(data);
       traverse();
    }
    return 0;
}
```

Status : Correct   ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Libin Mathew
Email: 241801137@rajalakshmi.edu.in
Roll no: 241801137
Phone: 9947119753
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Rohan is a software developer who is working on an application that processes data stored in a Doubly Linked List. He needs to implement a feature that finds and prints the middle element(s) of the list. If the list contains an odd number of elements, the middle element should be printed. If the list contains an even number of elements, the two middle elements should be printed.

Help Rohan by writing a program that reads a list of numbers, prints the list, and then prints the middle element(s) based on the number of elements in the list.

Input Format

The first line of the input consists of an integer n the number of elements in the doubly linked list.

The second line consists of n space-separated integers representing the elements of the list.

Output Format

The first line prints the elements of the list separated by space. (There is an extra space at the end of this line.)

The second line prints the middle element(s) based on the number of elements.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
20 52 40 16 18

Output: 20 52 40 16 18
40

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

void printList(struct Node* node) {
   while (node != NULL) {
      printf("%d ", node->data);
      node = node->next;
   }
   printf("\n");
}

void printMiddle(struct Node* head, int n) {
    struct Node* temp = head;
    if (n % 2 == 1) {
        for (int i = 0; i < n / 2; i++) {
            temp = temp->next;
        }
        printf("%d\n", temp->data);
    } else {
        for (int i = 0; i < (n / 2) - 1; i++) {
            temp = temp->next;
        }
        printf("%d %d\n", temp->data, temp->next->data);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int *arr = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < n; i++) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = arr[i];
        newNode->prev = tail;
        newNode->next = NULL;
        if (tail != NULL) {
            tail->next = newNode;
        }
        tail = newNode;
        if (head == NULL) {
            head = newNode;
        }
    }

    printList(head);
    printMiddle(head, n);

    // Free allocated memory
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }
    free(arr);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Tom is a software developer working on a project where he has to check if a doubly linked list is a palindrome. He needs to write a program to solve this problem. Write a program to help Tom check if a given doubly linked list is a palindrome or not.

Input Format

The first line 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 linked list elements.

Output Format

The first line displays the space-separated integers, representing the doubly linked list.

The second line displays one of the following:

1. If the doubly linked list is a palindrome, print "The doubly linked list is a palindrome".
2. If the doubly linked list is not a palindrome, print "The doubly linked list is not a palindrome".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 2 1

Output: 1 2 3 2 1
The doubly linked list is a palindrome

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

void printList(struct Node* node) {
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
    printf("\n");
}

int isPalindrome(struct Node* head, int n) {
    struct Node* start = head;
    struct Node* end = head;

    // Move 'end' to the last node
    while (end->next != NULL) {
        end = end->next;
    }

    for (int i = 0; i < n / 2; i++) {
        if (start->data != end->data) {
            return 0; // Not a palindrome
        }
        start = start->next;
        end = end->prev;
    }
    return 1; // Is a palindrome
}

int main() {
    int n;
    scanf("%d", &n);
    int *arr = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < n; i++) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = arr[i];
        newNode->prev = tail;
        newNode->next = NULL;
        if (tail != NULL) {
            tail->next = newNode;
        }
        tail = newNode;
        if (head == NULL) {
            head = newNode;
        }
    }

    printList(head);
    if (isPalindrome(head, n)) {
        printf("The doubly linked list is a palindrome\n");
    } else {
        printf("The doubly linked list is not a palindrome\n");
    }

    // Free allocated memory
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }
    free(arr);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Bala is a student learning about the doubly linked list and its functionalities. He came across a problem where he wanted to create a doubly linked list by appending elements to the front of the list.

After populating the list, he wanted to delete the node at the given position from the beginning. Write a suitable code to help Bala.

Input Format

The first line contains an integer N, the number of elements in the doubly linked list.

The second line contains N integers separated by a space, the data values of the nodes in the doubly linked list.

The third line contains an integer X, the position of the node to be deleted from the doubly linked list.

Output Format

The first line of output displays the original elements of the doubly linked list, separated by a space.

The second line prints the updated list after deleting the node at the given position X from the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
2
Output: 50 40 30 20 10
50 30 20 10

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

void printList(struct Node* node) {
   while (node != NULL) {
      printf("%d ", node->data);
      node = node->next;
   }
   printf("\n");
}

void deleteNodeAtPosition(struct Node** head_ref, int position) {
   if (*head_ref == NULL) {
      return;
   }

   struct Node* current = *head_ref;

   // If the head needs to be deleted
   if (position == 1) {
      *head_ref = current->next;
      if (*head_ref != NULL) {
         (*head_ref)->prev = NULL;
      }
      free(current);
      return;
   }
```

```
    // Traverse to the node at the given position
    for (int i = 1; current != NULL && i < position; i++) {
        current = current->next;
    }

    // If the position is more than the number of nodes
    if (current == NULL) {
        return;
    }

    // Update the next node's prev pointer if it exists
    if (current->next != NULL) {
        current->next->prev = current->prev;
    }

    // Update the previous node's next pointer if it exists
    if (current->prev != NULL) {
        current->prev->next = current->next;
    }

    free(current);
}

int main() {
    int n, x;
    scanf("%d", &n);
    int *arr = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    scanf("%d", &x);

    struct Node* head = NULL;
    for (int i = 0; i < n; i++) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = arr[i];
        newNode->prev = NULL;
        newNode->next = head;
        if (head != NULL) {
            head->prev = newNode;
        }
        head = newNode;
    }

    printList(head);
    deleteNodeAtPosition(&head, x);
    printList(head);

    // Free allocated memory
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }
    free(arr);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Riya is developing a contact management system where recently added contacts should appear first. She decides to use a doubly linked list to store contact IDs in the order they are added. Initially, new contacts are inserted at the front of the list. However, sometimes she needs to insert a new contact at a specific position in the list based on priority.

Help Riya implement this system by performing the following operations:

Insert contact IDs at the front of the list as they are added.Insert a new contact at a given position in the list.

Input Format

The first line of input consists of an integer N, representing the initial size of the linked list.

The second line consists of N space-separated integers, representing the values of the linked list to be inserted at the front.

The third line consists of an integer position, representing the position at which the new value should be inserted (position starts from 1).

The fourth line consists of integer data, representing the new value to be inserted.

Output Format

The first line of output prints the original list after inserting initial elements to the front.

The second line prints the updated linked list after inserting the element at the specified position.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 20 30 40
3
25

Output: 40 30 20 10
40 30 25 20 10

Answer

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

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

void printList(struct Node* node) {
   while (node != NULL) {
      printf("%d ", node->data);
      node = node->next;
   }
   printf("\n");
}
```

```
void insertAtFront(struct Node** head_ref, int new_data) {
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
   new_node->data = new_data;
   new_node->prev = NULL;
   new_node->next = (*head_ref);
   if ((*head_ref) != NULL) {
      (*head_ref)->prev = new_node;
   }
   (*head_ref) = new_node;
}

void insertAtPosition(struct Node** head_ref, int position, int new_data) {
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
   new_node->data = new_data;
   new_node->prev = NULL;
   new_node->next = NULL;

   if (position == 1) {
      new_node->next = *head_ref;
      if (*head_ref != NULL) {
         (*head_ref)->prev = new_node;
      }
      *head_ref = new_node;
      return;
   }

   struct Node* current = *head_ref;
   for (int i = 1; i < position - 1 && current != NULL; i++) {
      current = current->next;
   }

   if (current == NULL) {
      return;
   }

   new_node->next = current->next;
   new_node->prev = current;
   if (current->next != NULL) {
      current->next->prev = new_node;
   }
   current->next = new_node;
}

int main() {
   int n, position, data;
   scanf("%d", &n);
   int *arr = (int*)malloc(n * sizeof(int));
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }
   scanf("%d", &position);
   scanf("%d", &data);

   struct Node* head = NULL;
   for (int i = 0; i < n; i++) {
      insertAtFront(&head, arr[i]);
   }

   printList(head);
   insertAtPosition(&head, position, data);
   printList(head);

   // Free allocated memory
   struct Node* current = head;
   while (current != NULL) {
      struct Node* next = current->next;
      free(current);
      current = next;
   }
   free(arr);

   return 0;
}
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Pranav wants to clockwise rotate a doubly linked list by a specified number of positions. He needs your help to implement a program to achieve this. Given a doubly linked list and an integer representing the number of positions to rotate, write a program to rotate the list clockwise.

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 linked list elements.

The third line consists of an integer k, representing the number of places to rotate the list.

Output Format

The output displays the elements of the doubly linked list after rotating it by k positions.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
1

Output: 5 1 2 3 4

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

void printList(struct Node* node) {
   while (node != NULL) {
      printf("%d ", node->data);
      node = node->next;
   }
   printf("\n");
```

```
}

void rotateClockwise(struct Node** head_ref, int k) {
    if (*head_ref == NULL || k == 0) {
        return;
    }

    struct Node* current = *head_ref;
    int n = 1;
    while (current->next != NULL) {
        current = current->next;
        n++;
    }

    k = k % n;
    if (k == 0) {
        return;
    }

    current->next = *head_ref;
    (*head_ref)->prev = current;

    int steps = n - k;
    current = *head_ref;
    while (steps > 1) {
        current = current->next;
        steps--;
    }

    *head_ref = current->next;
    (*head_ref)->prev = NULL;
    current->next = NULL;
}

int main() {
    int n, k;
    scanf("%d", &n);
    int *arr = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    scanf("%d", &k);
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < n; i++) {
        struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
        new_node->data = arr[i];
        new_node->prev = tail;
        new_node->next = NULL;
        if (tail != NULL) {
            tail->next = new_node;
        }
        tail = new_node;
        if (head == NULL) {
            head = new_node;
        }
    }

    rotateClockwise(&head, k);
    printList(head);

    // Free allocated memory
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }
    free(arr);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Libin Mathew
Email: 241801137@rajalakshmi.edu.in
Roll no: 241801137
Phone: 9947119753
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

You are required to implement a program that deals with a doubly linked list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list.Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list.Display the List: Display the elements of the doubly linked list.

Input Format

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

The second line consists of n space-separated integers, denoting the elements to be inserted at the end.

The third line consists of integer m, representing the new element to be inserted.

The fourth line consists of an integer p, representing the position at which the new element should be inserted (1-based indexing).

Output Format

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints the original list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 25 34 48 57
35
4
Output: 10 25 34 35 48 57

Answer

-

Status : Skipped ***Marks : 0/10***

2. Problem Statement

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

Input Format

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

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

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8
Output: 8 4 6 10 12

Answer

-

Status : Skipped *Marks : 0/10*

3. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

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

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

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

Output: List in original order:
1 2 3 4 5
List in reverse order:
5 4 3 2 1

Answer

-

Status : Skipped ***Marks : 0/10***