Rajalakshmi Engineering College

Name: Sajine Santhakumar
Email: 240701459@rajalakshmi.edu.in
Roll no: 240701459
Phone: 9952076750
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 3
Total Mark : 30
Marks Obtained : 30

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

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

// Define the structure for a doubly linked list node
struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};

// Function to create a new node
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
```

```
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to insert a node at the end of the doubly linked list
void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

// Function to insert a node at a given position
void insertAtPosition(struct Node** head, int data, int position) {
    struct Node* newNode = createNode(data);
    if (position < 1) {
        return; // Invalid position
    }
    if (position == 1) {
        newNode->next = *head;
        if (*head != NULL) {
            (*head)->prev = newNode;
        }
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    for (int i = 1; i < position - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        return; // Invalid position
    }
    newNode->next = temp->next;
    if (temp->next != NULL) {
        temp->next->prev = newNode;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

// Function to display the elements of the doubly linked list
void displayList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Node* head = NULL;
    int n, m, p;

    // Input number of initial elements
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        int element;
        scanf("%d", &element);
        insertAtEnd(&head, element);
    }

    // Input new element and position
    scanf("%d", &m);
    scanf("%d", &p);

    // Insert at the specified position
    if (p < 1 || p > n + 1) {
        printf("Invalid position\n");
        displayList(head);
    } else {
        insertAtPosition(&head, m, p);
        displayList(head);
    }
    return 0;
}
```

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

2. Problem Statement

Vanessa is learning about the doubly linked list data structure and is eager to play around with it. She decides to find out how the elements are inserted at the beginning and end of the list.

Help her implement a program for the same.

Input Format

The first line of input contains an integer N, representing the size of the doubly linked list.

The next line contains N space-separated integers, each representing the values to be inserted into the doubly linked list.

Output Format

The first line of output prints the integers, after inserting them at the beginning, separated by space.

The second line prints the integers, after inserting at the end, separated by space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

Output: 5 4 3 2 1
1 2 3 4 5

Answer

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

// Node structure for doubly linked list
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};

// Function to insert a node at the beginning
void insertAtBeginning(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = *head;

   if (*head != NULL)
      (*head)->prev = newNode;

   *head = newNode;
}

// Function to insert a node at the end
void insertAtEnd(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;

   if (*head == NULL) {
      newNode->prev = NULL;
      *head = newNode;
      return;
   }

   struct Node* temp = *head;
   while (temp->next != NULL)
      temp = temp->next;

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

// Function to print the list from head to end
void printList(struct Node* head) {
   struct Node* temp = head;
   while (temp != NULL) {
      printf("%d ", temp->data);
      temp = temp->next;
   }
   printf("\n");
}

int main() {
   int n, value;
   scanf("%d", &n);

   struct Node* headBegin = NULL;
   struct Node* headEnd = NULL;

   for (int i = 0; i < n; i++) {
      scanf("%d", &value);
      insertAtBeginning(&headBegin, value);
      insertAtEnd(&headEnd, value);
   }

   printList(headBegin); // Output after inserting at beginning
   printList(headEnd);   // Output after inserting at end

   return 0;
}
```

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

3. 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

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

// Node structure for doubly linked list
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};

// Function to insert at the end of the doubly linked list
void insertEnd(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
```

```
    if (*head == NULL) {
        newNode->prev = NULL;
        *head = newNode;
        return;
    }

    struct Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;

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

// Function to print unique elements in reverse
void printUniqueReverse(struct Node* head) {
    int seen[101] = {0}; // Since values range from 1 to 100
    struct Node* temp = head;

    // Move to the last node
    while (temp && temp->next != NULL)
        temp = temp->next;

    // Traverse backward and print if not seen before
    while (temp != NULL) {
        if (!seen[temp->data]) {
            printf("%d ", temp->data);
            seen[temp->data] = 1;
        }
        temp = temp->prev;
    }
}

int main() {
    int n, value;
    scanf("%d", &n);

    struct Node* head = NULL;

    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        insertEnd(&head, value);
    }

    printUniqueReverse(head);

    return 0;
}
```

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