Rajalakshmi Engineering College

Name: NISHANTH ELANGO RAJAN
Email: 241901075@rajalakshmi.edu.in
Roll no: 241901075
Phone: 9444909050
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. 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

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

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

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

void insertEnd(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;
}

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

void printReverseList(struct Node* head) {
    if (head == NULL) {
        return;
    }
    struct Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

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

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

    printf("List in original order:\n");
    printList(head);

    printf("List in reverse order:\n");
    printReverseList(head);

    return 0;
}
```

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

2. Problem Statement

Aarav is working on a program to analyze his test scores, which are stored in a doubly linked list. He needs a solution to input scores into the list and determine the highest score.

Help him by providing code that lets users enter test scores into the doubly linked list and find the maximum score efficiently.

Input Format

The first line 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 score to be inserted.

Output Format

The output prints an integer, representing the highest score present in the list.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 4
89 71 2 70
```

Output: 89

Answer

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

struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Memory allocation failed\n");
      exit(EXIT_FAILURE);
   }
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

void insertEnd(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;
}

int findMax(struct Node* head) {
   if (head == NULL) {
      printf("List is empty\n");
      return -1;
   }
```

```
    int maxScore = head->data;
    struct Node* current = head->next;
    while (current != NULL) {
        if (current->data > maxScore) {
            maxScore = current->data;
        }
        current = current->next;
    }
    return maxScore;
}

void freeList(struct Node* head) {
    struct Node* current = head;
    struct Node* nextNode;
    while (current != NULL) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}

int main() {
    int n, score;
    struct Node* head = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &score);
        insertEnd(&head, score);
    }
    int maxScore = findMax(head);
    printf("%d\n", maxScore);
    freeList(head);

    return 0;
}
```

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

3. 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

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

```
#include <stdlib.h>

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

struct Node* head = NULL;
struct Node* tail = NULL;

void insertAtEnd(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = tail;
   newNode->next = NULL;

   if (head == NULL) {
      head = newNode;
      tail = newNode;
   } else {
      tail->next = newNode;
      tail = newNode;
   }
}

void insertAtPosition(int data, int position) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;

   if (position == 1) {
      newNode->prev = NULL;
      newNode->next = head;
      if (head != NULL) {
         head->prev = newNode;
      }
      head = newNode;
      if (tail == NULL)
      {
         tail = newNode;
      }
      return;
```

```
    }

    struct Node* current = head;
    int count = 1;

    while (current != NULL && count < position - 1) {
        current = current->next;
        count++;
    }

    if (current == NULL) {
        printf("Invalid position\n");
        return;
    }

    newNode->prev = current;
    newNode->next = current->next;

    if (current->next != NULL) {
        current->next->prev = newNode;
    } else {
        tail = newNode;
    }

    current->next = newNode;
}

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

int main() {
    int n, data, m, p;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        insertAtEnd(data);
    }
    scanf("%d", &m);
    scanf("%d", &p);
    struct Node* tempHead = head;
    insertAtPosition(m, p);
    if (head == tempHead){
        displayList();
    }else{
        if(head == NULL || head->next == NULL){
            printf("Invalid position\n");
            displayList();
        }else if(p > n + 1){
            printf("Invalid position\n");
            displayList();
        }else{
            displayList();
        }
    }


    return 0;
}
```

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