Rajalakshmi Engineering College

Name: Jaidev Arunachalam
Email: 241801099@rajalakshmi.edu.in
Roll no: 241801099
Phone: 9940254113
Branch: REC
Department: I AI & DS FB
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ashwin is tasked with developing a simple application to manage a list of items in a shop inventory using a doubly linked list. Each item in the inventory has a unique identification number. The application should allow users to perform the following operations:

Create a List of Items: Initialize the inventory with a given number of items. Each item will be assigned a unique number provided by the user and insert the elements at end of the list.

Delete an Item: Remove an item from the inventory at a specific position.

Display the Inventory: Show the list of items before and after deletion.

If the position provided for deletion is invalid (e.g., out of range), it should display an error message.

Input Format

The first line contains an integer n, representing the number of items to be initially entered into the inventory.

The second line contains n integers, each representing the unique identification number of an item separated by spaces.

The third line contains an integer p, representing the position of the item to be deleted from the inventory.

Output Format

The first line of output prints "Data entered in the list:" followed by the data values of each node in the doubly linked list before deletion.

If p is an invalid position, the output prints "Invalid position. Try again."

If p is a valid position, the output prints "After deletion the new list:" followed by the data values of each node in the doubly linked list after deletion.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
5
Output: Data entered in the list:

```
 node 1 : 1
 node 2 : 2
 node 3 : 3
 node 4 : 4
Invalid position. Try again.
```

Answer

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

241801099

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

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

   if (*head == NULL) {
      *head = newNode;
   } else {
      struct Node* temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newNode;
      newNode->prev = temp;
   }
}

// Function to delete a node at a specific position
void deleteAtPosition(struct Node** head, int position) {
   if (*head == NULL) {
      return; // List is empty
   }

   struct Node* temp = *head;

   // If position is out of bounds
   int count = 1;
   while (temp != NULL && count < position) {
      temp = temp->next;
      count++;
   }

   if (temp == NULL) {
      printf("Invalid position. Try again.\n");
```

241801099

```
        return;
    }

    // If node to be deleted is the head
    if (temp == *head) {
        *head = temp->next;
        if (*head != NULL) {
            (*head)->prev = NULL;
        }
    } else {
        // Adjust previous and next pointers
        if (temp->prev != NULL) {
            temp->prev->next = temp->next;
        }
        if (temp->next != NULL) {
            temp->next->prev = temp->prev;
        }
    }
    free(temp);
}

// Function to display the list
void displayList(struct Node* head) {
    struct Node* temp = head;
    int nodeCount = 1;
    while (temp != NULL) {
        printf("node %d : %d  ", nodeCount, temp->data);
        temp = temp->next;
        nodeCount++;
    }
    printf("\n");
}

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

    // Read the number of items to be inserted
    scanf("%d", &n);

    // Read the item identification numbers and insert them into the list
    for (i = 0; i < n; i++) {
```

241801099

```
        scanf("%d", &item);
        insertAtEnd(&head, item);
    }

    // Display the initial list
    printf("Data entered in the list: ");
    displayList(head);

    // Read the position to delete
    scanf("%d", &p);

    // Delete the item at the specified position
    if (p < 1 || p > n) {
        printf("Invalid position. Try again.\n");
    } else {
        deleteAtPosition(&head, p);
        printf("After deletion the new list: ");
        displayList(head);
    }

    return 0;
}
```

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

241801099 241801099 241801099 241801099