Rajalakshmi Engineering College

Name: vinnarasan j
Email: 241501245@rajalakshmi.edu.in
Roll no: 241501245
Phone: 7305999373
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are working on a text processing tool and need to implement a feature that allows users to insert characters at a specific position.

Implement a program that takes user inputs to create a singly linked list of characters and inserts a new character after a given index in the list.

Input Format

The first line of input consists of an integer N, representing the number of characters in the linked list.

The second line consists of a sequence of N characters, representing the linked list.

The third line consists of an integer index, representing the index(0-based) after

which the new character node needs to be inserted.

The fourth line consists of a character value representing the character to be inserted after the given index.

Output Format

If the provided index is out of bounds (larger than the list size):

1. The first line of output prints "Invalid index".
2. The second line prints "Updated list: " followed by the unchanged linked list values.

Otherwise, the output prints "Updated list: " followed by the updated linked list after inserting the new character after the given index.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
a b c d e
2
X
Output: Updated list: a b c X d e

Answer

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

// Define the structure for a node in the linked list
struct Node {
    char data;
    struct Node* next;
};

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

```
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1); // Exit if memory allocation fails.
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a node at the end of the linked list
void insertEnd(struct Node** head, char 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;
}

// Function to insert a character after a given index
void insertAfterIndex(struct Node** head, int index, char value) {
    if (*head == NULL && index > 0) {
        printf("Invalid index\n");
        printf("Updated list: \n");
        return;
    }

    struct Node* newNode = createNode(value);
    if (index == -1) { // Special case to insert at the beginning (before the first
node)
        newNode->next = *head;
        *head = newNode;
        return;
    }

    struct Node* curr = *head;
    int count = 0;
    while (curr != NULL && count < index) {
```

```
        curr = curr->next;
        count++;
    }

    if (curr == NULL) {
        printf("Invalid index\n");
    } else {
        newNode->next = curr->next;
        curr->next = newNode;
    }
}

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

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

    // Read the number of characters in the linked list
    scanf("%d", &n);
    getchar(); // Consume the newline character after reading the integer

    // Read the characters of the linked list and insert them
    for (int i = 0; i < n; i++) {
        scanf("%c", &data);
        insertEnd(&head, data);
        if (i < n - 1)
            getchar(); // Consume space, but not after the last character.
    }

    // Read the index and the character to insert
    scanf("%d", &index);
    scanf(" %c", &value); // Read the character, skipping any leading whitespace
```

```
    // Insert the character after the given index
    insertAfterIndex(&head, index, value);

    // Display the updated linked list
    printf("Updated list: ");
    displayList(head);

    return 0;
}
```

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