

# Rajalakshmi Engineering College

Name: Sri lokeshkaran. D  
Email: 240701527@rajalakshmi.edu.in  
Roll no:  
Phone: 8778475556  
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 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>
```

```
typedef struct Node {  
    char data;  
    struct Node* next;  
} Node;
```

```
Node* createNode(char data) {  
    Node* newNode = (Node*)malloc(sizeof(Node));  
    if (newNode == NULL) {  
        printf("Memory allocation failed\n");  
    }
```

```

        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

```

```

void insertEnd(Node** head, char data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
}

```

```

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

```

```

void insertAfterIndex(Node** head, int index, char data) {
    Node* temp = *head;
    int count = 0;

    while (temp != NULL && count < index) {
        temp = temp->next;
        count++;
    }

    if (temp == NULL) {
        printf("Invalid index\n");
        printf("Updated list: ");
        printList(*head);
    }
}

```

```

        return;
    }

    Node* newNode = createNode(data);
    newNode->next = temp->next;
    temp->next = newNode;

    printf("Updated list: ");
    printList(*head);
}

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

int main() {
    int N, index;
    char ch;
    Node* head = NULL;

    scanf("%d", &N);

    for (int i = 0; i < N; i++) {
        scanf(" %c", &ch);
        insertEnd(&head, ch);
    }

    scanf("%d", &index);
    scanf(" %c", &ch);

    insertAfterIndex(&head, index, ch);

    freeList(head);

    return 0;
}

```

**Status :** Correct

**Marks :** 10/10