Rajalakshmi Engineering College

Name: Wipro Tony A
Email: 240801389@rajalakshmi.edu.in
Roll no: 240801389
Phone: 9487910589
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters.Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

Input Format

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

Output Format

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -

Output: Forward Playlist: a b c
Backward Playlist: c b a

Answer

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

struct Node {
    char item;
    struct Node* next;
    struct Node* prev;
};

// You are using GCC
struct Node* tail ;
void insertAtEnd(struct Node** head, char item) {
   struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
   newnode->item = item ;
   newnode->next = NULL ;
   newnode->prev = NULL ;
   if(*head == NULL){
      *head = tail = newnode ;
   }
```

```
    else {
        tail->next=newnode ;
        newnode->prev=tail ;
        tail = newnode ;
    }
}
void displayForward(struct Node* head) {
    struct Node* temp = head ;
    while(temp!=NULL){
        printf("%c ",temp->item);
        temp = temp ->next;
    }
    printf("\n");
}

void displayBackward(struct Node* tail) {
    struct Node* temp = tail ;
    while(temp!=NULL){
        printf("%c ",temp->item);
        temp= temp->prev;
    }
    printf("\n");
}

void freePlaylist(struct Node* head) {
    struct Node* temp = head;
    while(temp!=NULL){
        Node* nextnode=temp->next;
        free(temp);
        temp = nextnode;
    }
    head = NULL ;
    tail = NULL ;
}

int main() {
    struct Node* playlist = NULL;
    char item;

    while (1) {
        scanf(" %c", &item);
        if (item == '-') {
            break;
```

```
        }
        insertAtEnd(&playlist, item);
    }

    struct Node* tail = playlist;
    while (tail->next != NULL) {
        tail = tail->next;
    }

    printf("Forward Playlist: ");
    displayForward(playlist);

    printf("Backward Playlist: ");
    displayBackward(tail);

    freePlaylist(playlist);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Wipro Tony A
Email: 240801389@rajalakshmi.edu.in
Roll no: 240801389
Phone: 9487910589
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n, representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155
Output: 163

Answer

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

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

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

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

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

int getMaxID(Node* head) {
    if (head == NULL)
        return -1;
    int max = head->data;
    Node* temp = head->next;
    while (temp != NULL) {
        if (temp->data > max)
            max = temp->data;
        temp = temp->next;
    }
    return max;
}

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

    Node* head = NULL;

    if (n == 0) {
        printf("Empty list!\n");
        return 0;
    }

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

    int maxID = getMaxID(head);
    if (maxID == -1)
        printf("Empty list!\n");
    else
        printf("%d\n", maxID);

    Node* current = head;
```

```
    while (current != NULL) {
        Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Wipro Tony A
Email: 240801389@rajalakshmi.edu.in
Roll no: 240801389
Phone: 9487910589
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 2
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>
struct node{
   char data;
   struct node* next;
};
struct node* create(char data){
   struct node* newn=(struct node*)malloc(sizeof(struct node));
   newn->data=data;
   newn->next=NULL;
   return newn;
```

```
}
void insert(struct node* head,int index,char newc){
    struct node* temp=head;
    int count=0;
    while(temp!=NULL && count<index){
        temp=temp->next;
        count++;
    }
    if(temp==NULL){
        printf("Invalid index\n");
        return;
    }
    struct node* newn=create(newc);
    newn->next=temp->next;
    temp->next=newn;
}
void print(struct node* head){
    struct node* temp=head;
    printf("Updated list:");
    while(temp!=NULL){
        printf(" %c",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
int main(){
    int n,index;
    char newc;
    struct node *head=NULL,*tail=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        char ch;
        scanf(" %c",&ch);
        struct node* newn=create(ch);
        if(head==NULL){
            head=newn;
            tail=newn;
        }
        else{
            tail->next=newn;
            tail=newn;
        }
```

```
    }
    scanf("%d",&index);
    scanf(" %c",&newc);
    if(index>n){
        printf("Invalid index\n");
        print(head);
    }
    else{
        insert(head,index,newc);
        print(head);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Wipro Tony A
Email: 240801389@rajalakshmi.edu.in
Roll no: 240801389
Phone: 9487910589
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: 10 20 30 40 50

Answer

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

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

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

void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
```

```
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

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

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

    Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int id;
        scanf("%d", &id);
        append(&head, id);
    }

    printList(head);

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

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Wipro Tony A
Email: 240801389@rajalakshmi.edu.in
Roll no: 240801389
Phone: 9487910589
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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>
```

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

Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

void displayList(Node* head, const char* message) {
    printf("%s", message);
    int count = 1;
    Node* temp = head;
    while (temp != NULL) {
        printf(" node %d : %d", count++, temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int deleteAtPosition(Node** head, int position, int size) {
    if (position < 1 || position > size)
        return 0;
```

```
    Node* temp = *head;
    int count = 1;

    if (position == 1) {
        *head = temp->next;
        if (*head != NULL)
            (*head)->prev = NULL;
        free(temp);
        return 1;
    }

    while (count < position) {
        temp = temp->next;
        count++;
    }

    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    if (temp->next != NULL)
        temp->next->prev = temp->prev;

    free(temp);
    return 1;
}

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

    Node* head = NULL;
    for (int i = 0; i < n; i++) {
        int item;
        scanf("%d", &item);
        append(&head, item);
    }

    int position;
    scanf("%d", &position);

    displayList(head, "Data entered in the list: ");

    if (!deleteAtPosition(&head, position, n)) {
```

```
        printf("Invalid position. Try again.\n");
    } else {
        displayList(head, "After deletion the new list: ");
    }

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

    return 0;
}
```

Status : Correct *Marks : 10/10*