Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The output prints the sum of the coefficients of the polynomials.

Sample Test Case

Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0

Output: 18

Answer

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

struct Node {
   int coeff;
   int expn;
   struct Node *next;
};

struct Node* createNode(int coeff, int expn) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->coeff = coeff;
   newNode->expn = expn;
   newNode->next = NULL;
   return newNode;
}

void insertTerm(struct Node** poly, int coeff, int expn) {
   struct Node* newNode = createNode(coeff, expn);
   if (*poly == NULL || (*poly)->expn < expn) {
        newNode->next = *poly;
        *poly = newNode;
    } else {
        struct Node* temp = *poly;
        while (temp->next != NULL && temp->next->expn > expn) {
            temp = temp->next;
        }
        if(temp->next != NULL && temp->next->expn == expn) {
            temp->next->coeff += coeff;
            free(newNode);
        }else {
            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
}

int sum(struct Node* poly) {
    int sum = 0;
    struct Node* temp = poly;
    while(temp != NULL) {
        sum += temp->coeff;
        temp = temp->next;
    }
    return sum;
}

int main() {
    struct Node* poly1 = NULL;
    struct Node* poly2 = NULL;

    int n,m;

    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        int coeff, expn;
        scanf("%d %d", &coeff, &expn);
        insertTerm(&poly1, coeff, expn);
    }

    scanf("%d", &m);
    for(int i = 0; i < m; i++) {
        int coeff, expn;
        scanf("%d %d", &coeff, &expn);
        insertTerm(&poly2, coeff, expn);
    }

    struct Node* sumPoly = NULL;

    struct Node* temp1=poly1;
    while (temp1 != NULL) {
        insertTerm(&sumPoly, temp1->coeff, temp1->expn);
        temp1 = temp1->next;
    }

    struct Node* temp2=poly2;
    while (temp2 != NULL) {
        insertTerm(&sumPoly, temp2->coeff, temp2->expn);
        temp2 = temp2->next;
    }

    printf("%d",sum(sumPoly));
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

Input Format

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

The second line consists of N space-separated elements of the linked list.

The third line consists of an integer x, representing the position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

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

void insert(int);
void display_List();
void deleteNode(int);

struct node {
   int data;
   struct node* next;
} *head = NULL, *tail = NULL;

void insert(int data) {
   struct node* newNode = (struct node*)malloc(sizeof(struct node));
   newNode->data = data;
   newNode->next = NULL;

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

```
    }
}

void display_List() {
    struct node* temp = head;
    if (temp == NULL) {
        return;
    }

    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void deleteNode(int x) {
    if (head == NULL) {
        printf("Invalid position. Deletion not possible.\n");
        return;
    }

    struct node* temp = head;
    if (x == 1) {
        head = temp->next;
        display_List();
        free(temp);
        return;
    }

    for (int i = 1; temp != NULL && i < x - 1; i++) {
        temp = temp->next;
    }

    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position. Deletion not possible.");
        return;
    }

    struct node* deleteIt = temp->next;
    temp->next = temp->next->next;
    if (temp->next == NULL) {
        tail = temp;
    }

    free(deleteIt);
    display_List();
}
```

```
int main() {
    int num_elements, element, pos_to_delete;

    scanf("%d", &num_elements);

    for (int i = 0; i < num_elements; i++) {
        scanf("%d", &element);
        insert(element);
    }

    scanf("%d", &pos_to_delete);

    deleteNode(pos_to_delete);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
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 : 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));
   newNode->data = data;
   newNode->next = NULL;
```

```
    return newNode;
}

void insertAfter(Node** head, int index, char value) {
    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: ");
        temp = *head;
        while (temp != NULL) {
            printf("%c ", temp->data);
            temp = temp->next;
        }
        printf("\n");
        return;
    }

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

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

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

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

    Node* head = NULL, *tail = NULL;
    for (int i = 0; i < n; i++) {
        char ch;
        scanf(" %c", &ch);
        Node* newNode = createNode(ch);
        if (head == NULL) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
    }

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

    Node* temp = head;
    int len = 0;
    while (temp != NULL) {
        temp = temp->next;
        len++;
    }

    if(index >= len) {
        printf("Invalid index\n");
        printf("Updated list: ");
        temp = head;
        while (temp != NULL) {
            printf("%c ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    } else {
        insertAfter(&head, index, value);
        printList(head);
    }

    freeList(head);
    return 0;

}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

As part of a programming assignment in a data structures course, students are required to create a program to construct a singly linked list by inserting elements at the beginning.

You are an evaluator of the course and guide the students to complete the task.

Input Format

The first line of input consists of an integer N, which is the number of elements.

The second line consists of N space-separated integers.

Output Format

The output prints the singly linked list elements, after inserting them at the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
78 89 34 51 67

Output: 67 51 34 89 78

Answer

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

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

void insertAtFront(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = *head;
   *head = newNode;
}

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

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

   int n;
   scanf("%d", &n);
```

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

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

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are tasked with developing a simple GPA management system using a singly linked list. The system allows users to input student GPA values, insertion should happen at the front of the linked list, delete record by position, and display the updated list of student GPAs.

Input Format

The first line of input contains an integer n, representing the number of students.

The next n lines contain a single floating-point value representing the GPA of each student.

The last line contains an integer position, indicating the position at which a student record should be deleted. Position starts from 1.

Output Format

After deleting the data in the given position, display the output in the format "GPA: " followed by the GPA value, rounded off to one decimal place.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3.8
3.2
3.5
4.1
2

Output: GPA: 4.1
GPA: 3.2
GPA: 3.8

Answer

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

typedef struct Node {
   float gpa;
   struct Node* next;
} Node;

void insertFront(Node** head, float gpa) {
   struct Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->gpa = gpa;
   newNode->next = *head;
   *head = newNode;
}

void deleteNode(Node** head, int i) {
   if (*head == NULL) return;

   Node* temp = *head;
   if (i == 1) {
       *head = temp->next;
       free(temp);
       return;
   }

   for (int j = 1; j < i - 1; j++){
       if (temp == NULL || temp->next == NULL)
           return;

       temp = temp->next;
   }

   if (temp == NULL || temp->next == NULL) return;

   Node* nodetoDelete = temp->next;
   temp->next = nodetoDelete->next;
   free(nodetoDelete);
}

void printList(Node* head) {
   Node* temp = head;
   while (temp != NULL) {
       printf("GPA: %.1f\n", temp->gpa);
       temp = temp->next;
   }
}

int main() {
   int n, i;
   float gpa;
   struct Node* head = NULL;

   scanf("%d", &n);

   for (int j = 0; j < n; j++) {
       scanf("%f", &gpa);
       insertFront(&head, gpa);
   }

   scanf("%d", &i);

   deleteNode(&head, i);
   printList(head);
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is tasked with creating a program to manage student roll numbers using a singly linked list.

Write a program for John that accepts students' roll numbers, inserts them at the end of the linked list, and displays the numbers.

Input Format

The first line of input consists of an integer N, representing the number of students.

The second line consists of N space-separated integers, representing the roll numbers of students.

Output Format

The output prints the space-separated integers singly linked list, after inserting the roll numbers of students at the end.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 85 47 62 31

Output: 23 85 47 62 31

Answer

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

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

struct Node* createNode(int rollNumber) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Memory allocation failed\n");
      exit(1);
   }
   newNode->data = rollNumber;
   newNode->next = NULL;
   return newNode;
}

void insertAtEnd(struct Node** head, int rollNumber) {
   struct Node* newNode = createNode(rollNumber);

   if(*head == NULL){
      *head = newNode;
      return;
   }

   struct Node* temp = *head;
```

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

void displayList(struct Node* head) {
    struct Node* temp = head;

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

    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}
int main() {
    struct Node* head = NULL;
    int N, rollNumber;
    scanf("%d", &N);
    for (int i = 0; i < N; i++){
        scanf("%d", &rollNumber);
        insertAtEnd(&head, rollNumber);
    }
    displayList(head);
    freeList(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukundan D
Email: 240801210@rajalakshmi.edu.in
Roll no: 240801210
Phone: 9080610536
Branch: REC
Department: I ECE FC
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Dev is tasked with creating a program that efficiently finds the middle element of a linked list. The program should take user input to populate the linked list by inserting each element into the front of the list and then determining the middle element.

Assist Dev, as he needs to ensure that the middle element is accurately identified from the constructed singly linked list:

If it's an odd-length linked list, return the middle element.If it's an even-length linked list, return the second middle element of the two elements.

Input Format

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

The second line consists of n space-separated integers, representing the elements of the list.

Output Format

The first line of output displays the linked list after inserting elements at the front.

The second line displays "Middle Element: " followed by the middle element of the linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
};


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

   return NULL;
}
```

```
Node* push(Node* head, int data) {
   Node* newNode = createNode(data);

   if (newNode == NULL) {
      printf("Memory allocation failed\n");
      exit(1);
   }

   newNode->next = head;
   return newNode;
}

int printMiddle(Node* head) {
   if (head != NULL) {
      Node* slow = head;
      Node* fast = head;

      while (fast != NULL && fast->next != NULL) {
         slow = slow->next;
         fast = fast->next->next;
      }

      return slow->data;
   }
}


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

   scanf("%d", &n);
   int value;

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

   struct Node* current = head;
   while (current != NULL) {
      printf("%d ", current->data);
```

```
        current = current->next;
    }
    printf("\n");


    int middle_element = printMiddle(head);
    printf("Middle Element: %d\n", middle_element);


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

    return 0;
}
```

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