Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Arrays and Functions

Attempt : 1
Total Mark : 5
Marks Obtained : 4

Section 1 : Coding

1. Problem Statement

Write a program that will read a Matrix (two-dimensional arrays) and print the sum of all elements of each row by passing the matrix to a function.

Function Signature: void calculateRowSum(int [ ][ ], int, int)

Input Format

The first line consists of an integer M representing the number of rows.

The second line consists of an integer N representing the number of columns.

The next M lines consist of N space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the sum of all elements of each row separated by a space.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
3
1 2 3
4 5 6
7 8 9
Output: 6 15 24

Answer

```
#include <stdio.h>

// You are using GCC
void calculateRowSum(int matrix[20][20], int rows, int cols){
   for(int i=0;i<rows;i++){
      int a=0;
      for(int j=0;j<cols;j++){
         a+=matrix[i][j];
         }
      printf("%d",a);
      printf("\t");
   }
}

int main() {
   int matrix[20][20];
   int r, c;

   scanf("%d", &r);
   scanf("%d", &c);

   for (int i = 0; i < r; i++) {
      for (int j = 0; j < c; j++) {
         scanf("%d", &matrix[i][j]);
      }
   }
```

```
    calculateRowSum(matrix, r, c);
    return 0;
}
```

Status : Correct ***Marks : 1/1***

2. Problem Statement

Saurabh is the manager of a growing tech company. He needs a program to record and analyze the monthly salaries of his employees. The program will take the number of employees and their respective salaries as input and then calculate the average salary, and find the highest and lowest salary among them.

Help Saurabh automate this task efficiently.

Input Format

The first line of input consists of an integer n, representing the number of employees.

The second line consists of n integers, where each integer represents the salary of an employee.

Output Format

The output prints n lines, where each line will display: "Employee i: "Salary

Where i is the employee number (starting from 1) and salary is the respective salary of that employee.

After that, print the average salary in the following format: "Average Salary: "average_salary

Where average_salary is the average salary of all employees, rounded to two decimal places.

Next, print the highest salary in the following format: "Highest Salary: "max_salary

Where max_salary is the highest salary among all employees.

Finally, print the lowest salary in the following format:"Lowest Salary: "min_salary

Where min_salary is the lowest salary among all employees.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
4000
3500
6000
2500
4500

Output: Employee 1: 4000
Employee 2: 3500
Employee 3: 6000
Employee 4: 2500
Employee 5: 4500

Average Salary: 4100.00
Highest Salary: 6000
Lowest Salary: 2500

Answer

```
#include<stdio.h>
int main(){
    int a;
    scanf("%d",&a);
    int arr[a];
    for(int i=0;i<a;i++){
        scanf("%d",&arr[i]);
    }
    float tot=0;
    for(int j=0;j<a;j++){
        printf("Employee %d: %d\n",j+1,arr[j]);
        tot+=arr[j];
    }
    float avg=tot/a;
    for(int i=0;i<a;i++){
        for(int j=i+1;j<a;j++){
            if(arr[i]>arr[j]){
                int min=arr[j];
                arr[j]=arr[i];
                arr[i]=min;
            }
        }
    }
    printf("Average salary: %.2f\n",avg);
    printf("Highest salary: %d\n",arr[a-1]);
    printf("Lowest salary: %d\n",arr[0]);
    return 0;
}
```

Status : Correct ***Marks : 1/1***

3. Problem Statement

Write a program that reads an integer 'n' and a square matrix of size 'n x n' from the user. The program should then set all the elements in the lower triangular part of the matrix (including the main diagonal) to zero using a function and display the resulting matrix.

Function Signature: void setZeros(int [ ][ ], int)

Input Format

The first line consists of an integer M representing the number of rows & columns.

The next M lines consist of M space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the matrix containing M space-separated elements in M lines where the lower triangular elements are replaced with zero.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
10 20 30
40 50 60
70 80 90
Output: 0 20 30
0 0 60
0 0 0

Answer

```
#include <stdio.h>

// You are using GCC
void setZeros(int arr[10][10], int n) {
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(j<=i){
                arr[i][j]=0;
            }
        }
    }
}

int main() {
    int arr1[10][10];
    int n;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &arr1[i][j]);
        }
    }

    setZeros(arr1, n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", arr1[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Status : Correct ***Marks : 1/1***

4. Problem Statement

Tim is creating a program to track and analyze student attendance. The program requires two inputs: the total number of students (n) and the total number of class sessions (m). The task is to design and populate an attendance matrix, 'matrix', representing the attendance record of each student for each session.

The program's specific objective is to determine whether the last student on the list attended an even or odd number of classes. This functionality will aid teachers in quickly evaluating the attendance habits of individual students.

Input Format

The first line of input consists of a positive integer n, representing the number of students.

The second line consists of a positive integer m, representing the number of class sessions.

The next n lines consist of m space-separated positive integers representing the number of classes attended by the student.

Output Format

The output displays one of the following results:

If the last session is even the output prints "[LastSession] is even".

If the last session is odd the output prints "[LastSession] is odd".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
2
1 2
3 100
Output: 100 is even

Answer

```
#include<stdio.h>
int main(){
   int n,m,a;
   scanf("%d\n%d",&n,&m);
   int matrix[n][m];
   for(int i=0;i<n;i++){
      for(int j=0;j<m;j++){
         scanf("%d",&matrix[i][j]);
      }
   }
   a=matrix[n-1][m-1];
   if(a%2==0){
      printf("%d is even",a);
   }
   else
   printf("%d is odd",a);
}
```

Status : Correct ***Marks : 1/1***

5. Problem Statement

Alex, a budding programmer, is tasked with writing a menu-driven program to perform operations on an array of integers. The operations include finding the smallest number, the largest number, the sum of all numbers, and their average. The program must repeatedly display the menu until Alex chooses to exit.

Write a program to ensure the specified tasks are implemented based on Alex's choices.

Input Format

The first line contains an integer n, representing the number of elements in the array.

The second line contains n space-separated integers representing the array elements.

The subsequent lines contain integers representing the menu choices:

Choice 1: Find and display the smallest number.

Choice 2: Find and display the largest number.

Choice 3: Calculate and display the sum of all numbers.

Choice 4: Calculate and display the average of all numbers as double.

Choice 5: Exit the program.

Output Format

For each valid menu choice, print the corresponding result:

For choice 1, print "The smallest number is: X", where X is the smallest number in the array.

For choice 2, print "The largest number is: X", where X is the largest number in the array.

For choice 3, print "The sum of the numbers is: X", where X is the sum of all numbers in the array.

For choice 4, print "The average of the numbers is: X. XX", where X.XX is the double value representing an average of all numbers in the array, rounded to two decimal places.

For choice 5, print "Exiting the program".

If an invalid choice is made, print "Invalid choice! Please enter a valid option (1-5)."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
10 20 30
1
5

Output: The smallest number is: 10
Exiting the program

Answer

-

Status : Skipped ***Marks : 0/1***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : MCQ

1. Consider the singly linked list: 13 -> 4 -> 16 -> 9 -> 22 -> 45 -> 5 -> 16 -> 6, and an integer K = 10, you need to delete all nodes from the list that are less than the given integer K.

What will be the final linked list after the deletion?

Answer

13 -&gt; 16 -&gt; 22 -&gt; 45 -&gt; 16

Status : Correct ***Marks : 1/1***

2. Which of the following statements is used to create a new node in a singly linked list?

struct node {

```
    int data;
    struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Answer

ptr = (NODE*)malloc(sizeof(NODE));

Status : Correct ***Marks : 1/1***

3. Linked lists are not suitable for the implementation of?

Answer

Binary search

Status : Correct ***Marks : 1/1***

4. Given the linked list: 5 -> 10 -> 15 -> 20 -> 25 -> NULL. What will be the output of traversing the list and printing each node's data?

Answer

5 10 15 20 25

Status : Correct ***Marks : 1/1***

5. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list?

```
struct node {
    int data;
    struct node* next;
};
static void reverse(struct node** head_ref) {
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL) {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}
```

Answer

*head_ref = prev;

Status : Correct ***Marks : 1/1***

6. Consider an implementation of an unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operations can be implemented in O(1) time?

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

I and III

Status : Correct ***Marks : 1/1***

7. Given a pointer to a node X in a singly linked list. If only one point is given and a pointer to the head node is not given, can we delete node X from the given linked list?

Answer

Possible if X is not last node.

Status : Correct ***Marks : 1/1***

8. In a singly linked list, what is the role of the "tail" node?

Answer

It stores the last element of the list

Status : Correct ***Marks : 1/1***

9. The following function takes a singly linked list of integers as a parameter and rearranges the elements of the lists.

The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node* next;
};

void rearrange (struct node* list) {
   struct node *p,q;
   int temp;
   if (! List || ! list->next) return;
   p=list; q=list->next;
   while(q) {
      temp=p->value; p->value=q->value;
      q->value=temp;p=q->next;
      q=p?p->next:0;
   }
}
```

Answer

2, 1, 4, 3, 6, 5, 7

Status : Correct ***Marks : 1/1***

10. Consider the singly linked list: 15 -> 16 -> 6 -> 7 -> 17. You need to delete all nodes from the list which are prime.

What will be the final linked list after the deletion?

Answer

15 -&gt; 16 -&gt; 6

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. What is the primary advantage of using an array-based stack with a fixed size?

Answer

Efficient memory usage

Status : Correct ***Marks : 1/1***

2. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
void push(int* stack, int* top, int item) {
  if (*top == MAX_SIZE - 1) {
    printf("Stack Overflow\n");
```

```
        return;
    }
    stack[++(*top)] = item;
}
int pop(int* stack, int* top) {
    if (*top == -1) {
        printf("Stack Underflow\n");
        return -1;
    }
    return stack[(*top)--];
}

int main() {
    int stack[MAX_SIZE];
    int top = -1;
    push(stack, &top, 10);
    push(stack, &top, 20);
    push(stack, &top, 30);
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    printf("%d\n", pop(stack, &top));
    return 0;
}
```

Answer

302010Stack Underflow-1

Status : Correct ***Marks : 1/1***

3. Which of the following operations allows you to examine the top element of a stack without removing it?

Answer

Peek

Status : Correct ***Marks : 1/1***

4. In an array-based stack, which of the following operations can result in a Stack underflow?

Answer

Popping an element from an empty stack

Status : Correct ***Marks : 1/1***

5. Pushing an element into the stack already has five elements. The stack size is 5, then the stack becomes

Answer

Overflow

Status : Correct ***Marks : 1/1***

6. What is the value of the postfix expression 6 3 2 4 + - *?

Answer

-18

Status : Correct ***Marks : 1/1***

7. Consider a linked list implementation of stack data structure with three operations:

push(value): Pushes an element value onto the stack.pop(): Pops the top element from the stack.top(): Returns the item stored at the top of the stack.

Given the following sequence of operations:

push(10);pop();push(5);top();

What will be the result of the stack after performing these operations?

Answer

The top element in the stack is 5

Status : Correct ***Marks : 1/1***

8. In a stack data structure, what is the fundamental rule that is followed for performing operations?

Answer

Last In First Out

Status : Correct ***Marks : 1/1***

9. What is the advantage of using a linked list over an array for implementing a stack?

Answer

Linked lists can dynamically resize

Status : Correct ***Marks : 1/1***

10. A user performs the following operations on stack of size 5 then which of the following is correct statement for Stack?

```
push(1);
pop();
push(2);
push(3);
pop();
push(2);
pop();
pop();
push(4);
pop();
pop();
push(5);
```

Answer

Underflow Occurs

Status : Correct ***Marks : 1/1***

11. Which of the following Applications may use a Stack?

Answer

All of the mentioned options

Status : Correct *Marks : 1/1*

12. Elements are Added on _______ of the Stack.

Answer

Top

Status : Correct *Marks : 1/1*

13. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
void display() {
   if (top == -1) {
      printf("Stack is empty\n");
   } else {
      printf("Stack elements: ");
      for (int i = top; i >= 0; i--) {
         printf("%d ", stack[i]);
      }
      printf("\n");
   }
}
void push(int value) {
   if (top == MAX_SIZE - 1) {
      printf("Stack Overflow\n");
   } else {
      stack[++top] = value;
   }
}
int main() {
   display();
   push(10);
   push(20);
   push(30);
   display();
   push(40);
   push(50);
   push(60);
   display();
   return 0;
}
```

Answer

Stack is emptyStack elements: 30 20 10Stack OverflowStack elements: 50 40 30 20 10&nbsp;

Status : Correct ***Marks : 1/1***

14. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
int stack[MAX_SIZE];
int top = -1;
int isEmpty() {
    return (top == -1);
}
int isFull() {
    return (top == MAX_SIZE - 1);
}
void push(int item) {
    if (isFull())
        printf("Stack Overflow\n");
    else
        stack[++top] = item;
}
int main() {
    printf("%d\n", isEmpty());
    push(10);
    push(20);
    push(30);
    printf("%d\n", isFull());
    return 0;
}
```

Answer

10

Status : Correct ***Marks : 1/1***

15. Consider the linked list implementation of a stack.

Which of the following nodes is considered as Top of the stack?

Answer

First node

Status : Correct ***Marks : 1/1***

16. The user performs the following operations on the stack of size 5 then at the end of the last operation, the total number of elements present in the stack is

```
push(1);
pop();
push(2);
push(3);
pop();
push(4);
pop();
pop();
push(5);
```

Answer

1

Status : Correct ***Marks : 1/1***

17. When you push an element onto a linked list-based stack, where does the new element get added?

Answer

At the beginning of the list

Status : Correct ***Marks : 1/1***

18. The result after evaluating the postfix expression 10 5 + 60 6 / * 8 - is

Answer

142

Status : Correct ***Marks : 1/1***

19. In the linked list implementation of the stack, which of the following operations removes an element from the top?

Answer

Peek

Status : Wrong ***Marks : 0/1***

20. Here is an Infix Expression: 4+3*(6*3-12). Convert the expression from Infix to Postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression?

Answer

4

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. Find the preorder traversal of the given binary search tree.

Answer

9, 2, 1, 6, 4, 7, 10, 14

Status : Correct ***Marks : 1/1***

2. In a binary search tree with nodes 18, 28, 12, 11, 16, 14, 17, what is the value of the left child of the node 16?

Answer

14

Status : Correct ***Marks : 1/1***

3. Find the postorder traversal of the given binary search tree.

Answer

1, 4, 2, 18, 14, 13

Status : Correct *Marks : 1/1*

4. Which of the following is a valid preorder traversal of the binary search tree with nodes: 18, 28, 12, 11, 16, 14, 17?

Answer

18, 12, 11, 16, 14, 17, 28

Status : Correct *Marks : 1/1*

5. Find the in-order traversal of the given binary search tree.

Answer

1, 2, 4, 13, 14, 18

Status : Correct *Marks : 1/1*

6. How many distinct binary search trees can be created out of 4 distinct keys?

Answer

14

Status : Correct *Marks : 1/1*

7. While inserting the elements 5, 4, 2, 8, 7, 10, 12 in a binary search tree, the element at the lowest level is ________.

Answer

12

Status : Correct ***Marks : 1/1***

8. The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19. Which one of the following is the postorder traversal of the tree?

Answer

11, 12, 10, 16, 19, 18, 20, 15

Status : Correct ***Marks : 1/1***

9. While inserting the elements 71, 65, 84, 69, 67, 83 in an empty binary search tree (BST) in the sequence shown, the element in the lowest level is _______.

Answer

67

Status : Correct ***Marks : 1/1***

10. Find the pre-order traversal of the given binary search tree.

Answer

13, 2, 1, 4, 14, 18

Status : Correct ***Marks : 1/1***

11. Find the post-order traversal of the given binary search tree.

Answer

10, 17, 20, 18, 15, 32, 21

Status : Correct ***Marks : 1/1***

12. Which of the following operations can be used to traverse a Binary Search Tree (BST) in ascending order?

Answer

Inorder traversal

Status : Correct ***Marks : 1/1***

13. Which of the following is the correct post-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

20, 32, 30, 52, 57, 55, 50

Status : Correct ***Marks : 1/1***

14. Which of the following is the correct pre-order traversal of a binary search tree with nodes: 50, 30, 20, 55, 32, 52, 57?

Answer

50, 30, 20, 32, 55, 52, 57

Status : Correct ***Marks : 1/1***

15. Which of the following is the correct in-order traversal of a binary search tree with nodes: 9, 3, 5, 11, 8, 4, 2?

Answer

2, 3, 4, 5, 8, 9, 11

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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>

typedef struct poly
{
   int coeff;
   int expon;
   struct poly*next;
}node;

node* newnode(int coeff,int expon)
{
   node* newNode=(node*)malloc(sizeof(node));
   newNode->coeff=coeff;
   newNode->expon=expon;
   newNode->next=NULL;
   return newNode;
}

void insertNode(node** head,int coeff,int expon)
```

```
{
   node* temp=*head;
   if(temp==NULL)
   {
      *head=newnode(coeff,expon);
      return;
   }
   while(temp->next!=NULL)
   {
      temp=temp->next;
   }
   temp->next=newnode(coeff,expon);
}

int main()
{
   int n,coeff,expon;
   scanf("%d",&n);
   node* poly1;
   node* poly2;
   for(int i=0;i<n;i++)
   {
      scanf("%d %d",&coeff,&expon);
      insertNode(&poly2,coeff,expon);
   }
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      scanf("%d %d",&coeff,&expon);
      insertNode(&poly2,coeff,expon);
   }
   int sum=0;
   while(poly1!=NULL)
   {
      sum+=poly1->coeff;
      poly1=poly1->next;
   }
   while(poly2!=NULL)
   {
      sum+=poly2->coeff;
      poly2=poly2->next;
   }
```

```
    printf("%d",sum);
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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 : 9

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 value){
struct node*newn=(struct node*)malloc(sizeof(struct node));
newn->data=value;
newn->next=NULL;
if(head==NULL){
head=newn;
tail=newn;
}
else{
   tail->next=newn;
```

```
        tail=newn;
    }
}
void display_List(){
    struct node *temp=head;
    if(temp==NULL){
        printf("List is empty\n");
        return;
}
while(temp!=NULL){
    printf("%d",temp->data);
    temp=temp->next;
}
printf("\n");
}
void deleteNode(int pos){
    if(head==NULL){
        printf("Invalid position.Deletion not possible.\n");
        return;
    }
    struct node*temp=head;
    if(pos==1){
        head=head->next;
        free(temp);
        display_List();
        return;
    }
    struct node*prev=NULL;
    int count=1;
    while(temp!=NULL && count<pos){
        prev=temp;
        temp=temp->next;
        count++;

    }
    if(temp==NULL){
        printf("Invalid postion. Deletion not possible.\n");
        return;
    }
    prev->next=temp->next;
    if(temp==tail){
        tail=prev;
    }
    free(temp);
    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 : Partially correct ***Marks : 9/10***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
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>
#include <string.h>

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

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

```
    if (!newNode) {
        printf("Memory error!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Append character to the end of the list
void append(Node** head, char data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next)
        temp = temp->next;
    temp->next = newNode;
}

// Insert character after given index
int insertAfterIndex(Node* head, int index, char data) {
    Node* temp = head;
    int count = 0;

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

    if (!temp) {
        return 0;  // Invalid index
    }

    Node* newNode = createNode(data);
    newNode->next = temp->next;
    temp->next = newNode;
    return 1;  // Success
}
```

```
// Print the linked list
void printList(Node* head) {
    printf("Updated list: ");
    Node* temp = head;
    while (temp) {
        printf("%c", temp->data);
        if (temp->next)
            printf(" ");
        temp = temp->next;
    }
    printf("\n");
}

// Free the list
void freeList(Node* head) {
    Node* temp;
    while (head) {
        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);
        append(&head, ch);
    }

    scanf("%d", &index);
    char insertChar;
    scanf(" %c", &insertChar);

    int success = insertAfterIndex(head, index, insertChar);

    if (!success)
        printf("Invalid index\n");
    printList(head);
    freeList(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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 value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = *head;
   *head = newNode;
}
void printList(struct Node* head) {
   struct Node* current = head;
   while (current != NULL) {
      printf("%d ", current->data);
      current = current->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: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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>

// Node structure for GPA
struct Node {
   float gpa;
   struct Node* next;
};

// Insert GPA at front
void insertAtFront(struct Node** head, float gpa) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->gpa = gpa;
   newNode->next = *head;
   *head = newNode;
}

// Delete node at given position (1-based)
void deleteAtPosition(struct Node** head, int position) {
   if (*head == NULL || position < 1) return;
```

```
    struct Node* temp = *head;

    // Delete head
    if (position == 1) {
        *head = temp->next;
        free(temp);
        return;
    }

    // Traverse to the node before the one to delete
    for (int i = 1; temp != NULL && i < position - 1; i++)
        temp = temp->next;

    // If position is invalid
    if (temp == NULL || temp->next == NULL)
        return;

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

// Print the GPA list
void printGPAList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("GPA: %.1f\n", current->gpa);
        current = current->next;
    }
}

// Free the memory
void freeList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }
}
```

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

    struct Node* head = NULL;
    float gpa;

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

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

    deleteAtPosition(&head, position);
    printGPAList(head);
    freeList(head);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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>

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

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

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

// Function to print the list
void printList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

// Free memory
void freeList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }
}

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

    struct Node* head = NULL;

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

    printList(head);
    freeList(head);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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;
};


struct Node* push(struct Node* head, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = head;
   return newNode;
}
int printMiddle(struct Node* head) {
   struct Node* slow = head;
   struct 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*

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a coding competition, you are assigned a task to create a program that simulates a stack using a linked list.

The program should feature a menu-driven interface for pushing an integer to stack, popping, and displaying stack elements, with robust error handling for stack underflow situations. This challenge tests your data structure skills.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the integer value onto the stack. If the choice is 1, the following input is a space-separated integer, representing the element to be pushed onto the stack.

Choice 2: Pop the integer from the stack.

Choice 3: Display the elements in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

If the choice is 1, push the given integer to the stack and display the following: "Pushed element: " followed by the value pushed.

If the choice is 2, pop the integer from the stack and display the following: "Popped element: " followed by the value popped.

If the choice is 2, and if the stack is empty without any elements, print "Stack is empty. Cannot pop."

If the choice is 3, print the elements in the stack: "Stack elements (top to bottom): " followed by the space-separated values.

If the choice is 3, and there are no elements in the stack, print "Stack is empty".

If the choice is 4, exit the program and display the following: "Exiting program".

If any other choice is entered, print "Invalid choice".

Refer to the sample input and output for the exact format.

Sample Test Case

Input: 1 3
1 4
3
2
3
4
Output: Pushed element: 3
Pushed element: 4
Stack elements (top to bottom): 4 3
Popped element: 4
Stack elements (top to bottom): 3
Exiting program

Answer

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

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

struct Node* top = NULL;

void push(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (!newNode) {
      printf("Memory allocation failed\n");
      return;
   }
   newNode->data = value;
   newNode->next = top;
   top = newNode;
```

```
    printf("Pushed element: %d\n", value);
}

void pop() {
    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
        return;
    }
    struct Node* temp = top;
    printf("Popped element: %d\n", top->data);
    top = top->next;
    free(temp);
}

void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct Node* temp = top;
    printf("Stack elements (top to bottom):\n");
    while (temp != NULL) {
        printf("%d\n", temp->data);
        temp = temp->next;
    }
}

int main() {
    int choice, value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
```

```
            printf("Exiting program\n");
            return 0;
        default:
            printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 2.5

Section 1 : Coding

1. Problem Statement

Sanjeev is in charge of managing a library's book storage, and he wants to create a program that simplifies this task. His goal is to implement a program that simulates a stack using an array.

Help him in writing a program that provides the following functionality:

Add Book ID to the Stack (Push): You can add a book ID to the top of the book stack. Remove Book ID from the Stack (Pop): You can remove the top book ID from the stack and display its details. If the stack is empty, you cannot remove any more book IDs.Display Books ID in the Stack (Display): You can view the books ID currently on the stack.Exit the Library: You can choose to exit the program.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the book onto the stack. If the choice is 1, the following input is a space-separated integer, representing the ID of the book to be pushed onto the stack.

Choice 2: Pop the book ID from the stack.

Choice 3: Display the book ID in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, push the given book ID to the stack and display the corresponding message.
2. If the choice is 2, pop the book ID from the stack and display the corresponding message.
3. If the choice is 2, and if the stack is empty without any book ID, print "Stack Underflow"
4. If the choice is 3, print the book IDs in the stack.
5. If the choice is 3, and there are book IDs in the stack, print "Stack is empty"
6. If the choice is 4, exit the program and display the corresponding message.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1 19
1 28
2
3
2
4
Output: Book ID 19 is pushed onto the stack
Book ID 28 is pushed onto the stack

```
Book ID 28 is popped from the stack
Book ID in the stack: 19
Book ID 19 is popped from the stack
Exiting the program
```

Answer

```
// You are using GCC
#include <stdio.h>
#define MAX_SIZE 100

int stack[MAX_SIZE];
int top = -1;

void push(int book_id) {
    if (top < MAX_SIZE - 1) {
        stack[++top] = book_id;
        printf("Book ID %d is pushed onto the stack\n", book_id);
    } else {
        printf("Stack Overflow\n");
    }
}

void pop() {
    if (top >= 0) {
        printf("Book ID %d is popped from the stack\n", stack[top--]);
    } else {
        printf("Stack Underflow\n");
    }
}

void display() {
    if (top >= 0) {
        printf("Book ID in the stack:");
        for (int i = 0; i <= top; i++) {
            printf(" %d", stack[i]);
        }
        printf("\n");
    } else {
        printf("Stack is empty\n");
    }
}
```

```
int main() {
    int choice, book_id;

    while (1) {
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                scanf("%d", &book_id);
                push(book_id);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting the program\n");
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
}
```

Status : Partially correct ***Marks : 2.5/10***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Sharon is developing a programming challenge for a coding competition. The challenge revolves around implementing a character-based stack data structure using an array.

Sharon's project involves a stack that can perform the following operations:

Push a Character: Users can push a character onto the stack.Pop a Character: Users can pop a character from the stack, removing and displaying the top character.Display Stack: Users can view the current elements in the stack.Exit: Users can exit the stack operations application.

Write a program to help Sharon to implement a program that performs the given operations.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the character to be pushed onto the stack.

Choice 2: Pop the character from the stack.

Choice 3: Display the characters in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, push the given character to the stack and display the pushed character having the prefix "Pushed: ".
2. If the choice is 2, undo the character from the stack and display the character that is popped having the prefix "Popped: ".
3. If the choice is 2, and if the stack is empty without any characters, print "Stack is empty. Nothing to pop."
4. If the choice is 3, print the elements in the stack having the prefix "Stack elements: ".
5. If the choice is 3, and there are no characters in the stack, print "Stack is empty."
6. If the choice is 4, exit the program.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
4
Output: Stack is empty. Nothing to pop.

Answer

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

```
#include <stdbool.h>

#define MAX_SIZE 100

char items[MAX_SIZE];
int top = -1;

void initialize() {
   top = -1;
}
bool isFull() {
   return top == MAX_SIZE - 1;
}

bool isEmpty() {
   return top == -1;
}

void push(char value) {
   if (isFull()) {
      printf("Stack Overflow\n");
      return;
   }
   items[++top] = value;
   printf("Pushed: %c\n", value);
}

char pop() {
   if (isEmpty()) {
      printf("Stack is empty. Nothing to pop.\n");
      return '\0';
   }
   char value = items[top--];
   printf("Popped: %c\n", value);
   return value;
}

void display() {
   if (isEmpty()) {
      printf("Stack is empty.\n");
      return;
   }
   printf("Stack elements: ");
```

```
    for (int i = top; i >= 0; i--) {
        printf("%c\n", items[i]);
    }
}

int main() {
    initialize();
    int choice;
    char value;

    while (true) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are a software developer tasked with building a module for a scientific calculator application. The primary function of this module is to convert infix mathematical expressions, which are easier for users to read and write, into postfix notation (also known as Reverse Polish Notation). Postfix notation is more straightforward for the application to evaluate because it removes the need for parentheses and operator precedence rules.

The scientific calculator needs to handle various mathematical expressions with different operators and ensure the conversion is correct. Your task is to implement this infix-to-postfix conversion algorithm using a stack-based approach.

Example

Input:

a+b

Output:

ab+

Explanation:

The postfix representation of (a+b) is ab+.

Input Format

The input is a string, representing the infix expression.

Output Format

The output displays the postfix representation of the given infix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a+(b*e)
Output: abe*+

Answer

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

struct Stack {
   int top;
   unsigned capacity;
   char* array;
};

struct Stack* createStack(unsigned capacity) {
   struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));

   if (!stack)
```

```
        return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (char*)malloc(stack->capacity * sizeof(char));

    return stack;
}

int isEmpty(struct Stack* stack) {
    return stack->top == -1;
}

char peek(struct Stack* stack) {
    return stack->array[stack->top];
}

char pop(struct Stack* stack) {
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack* stack, char op) {
    stack->array[++stack->top] = op;
}
int isOperand(char ch) {
    return (ch >= 'a' && ch <= 'z') ||
           (ch >= 'A' && ch <= 'Z') ||
           (ch >= '0' && ch <= '9');
}
int Prec(char ch) {
    switch (ch) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
    }
    return -1;
}

void infixToPostfix(char* exp) {
    struct Stack* stack = createStack(strlen(exp));
    int i, k;

    for (i = 0, k = -1; exp[i]; ++i) {
        if (isOperand(exp[i])) {
            exp[++k] = exp[i];
        }
        else if (exp[i] == '(') {
            push(stack, exp[i]);
        }
        else if (exp[i] == ')') {
            while (!isEmpty(stack) && peek(stack) != '(')
                exp[++k] = pop(stack);
            if (!isEmpty(stack) && peek(stack) != '(')
                return;
            else
                pop(stack);
        }
        else {
            while (!isEmpty(stack) && Prec(exp[i]) <= Prec(peek(stack)))
                exp[++k] = pop(stack);
            push(stack, exp[i]);
        }
    }

    while (!isEmpty(stack))
        exp[++k] = pop(stack);

    exp[++k] = '\0';
    printf("%s\n", exp);
}

int main() {
    char exp[100];
    scanf("%s", exp);

    infixToPostfix(exp);
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Milton is a diligent clerk at a school who has been assigned the task of managing class schedules. The school has various sections, and Milton needs to keep track of the class schedules for each section using a stack-based system.

He uses a program that allows him to push, pop, and display class schedules for each section. Milton's program uses a stack data structure, and each class schedule is represented as a character. Help him write a program using a linked list.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the class schedule to be pushed onto the stack.

Choice 2: Pop class schedule from the stack

Choice 3: Display the class schedules in the stack.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the stack:

- If the choice is 1, push the given class schedule to the stack and display the following: "Adding Section: [class schedule]"
- If the choice is 2, pop the class schedule from the stack and display the following: "Removing Section: [class schedule]"
- If the choice is 2, and if the stack is empty without any class schedules, print "Stack is empty. Cannot pop."
- If the choice is 3, print the class schedules in the stack in the following: "Enrolled Sections: " followed by the class schedules separated by space.
- If the choice is 3, and there are no class schedules in the stack, print "Stack is empty"
- If the choice is 4, exit the program and display the following: "Exiting the program"
- If any other choice is entered, print "Invalid choice"

Refer to the sample output for the exact format.

Sample Test Case

Input: 1 d
1 h
3
2

3
4
Output: Adding Section: d
Adding Section: h
Enrolled Sections: h d
Removing Section: h
Enrolled Sections: d
Exiting program

Answer

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

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

struct Node* top = NULL;

void push(char value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = top;
   top = newNode;
   printf("Adding Section: %c\n", value);
}

void pop() {
   if (top == NULL) {
      printf("Stack is empty. Cannot pop.\n");
      return;
   }
   printf("Removing Section: %c\n", top->data);
   struct Node* temp = top;
   top = top->next;
   free(temp);
}

void displayStack() {
   if (top == NULL) {
      printf("Stack is empty\n");
```

```
        return;
    }
    printf("Enrolled Sections: ");
    struct Node* temp = top;
    while (temp != NULL) {
        printf("%c ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int choice;
    char value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf(" %c", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
                printf("Exiting program\n");
                break;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

John is learning about Binary Search Trees (BST) in his computer science class. He wants to create a program that allows users to delete a node with a given value from a BST and print the remaining nodes using an in-order traversal.

Implement a function to help him delete a node with a given value from a BST.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the BST nodes.

The third line consists of an integer V, which is the value to delete from the BST.

Output Format

The output prints the space-separated values in the BST in an in-order traversal, after the deletion of the specified value.

If the specified value is not available in the tree, print the given input values in-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7
15
Output: 2 5 7 10

Answer

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

struct TreeNode {
   int data;
   struct TreeNode* left;
   struct TreeNode* right;
};

struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
   newNode->data = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}
#include <stdio.h>
#include <stdlib.h>
```

```
// Define the structure for a BST node
typedef struct TreeNode {
    int key;
    struct TreeNode *left, *right;
} TreeNode;

// Function to create a new node
TreeNode* createNode(int key) {
    TreeNode* node = (TreeNode*)malloc(sizeof(TreeNode));
    node->key = key;
    node->left = node->right = NULL;
    return node;
}

// Function to insert a node into BST
TreeNode* insert(TreeNode* root, int key) {
    if (!root) return createNode(key);
    if (key < root->key) root->left = insert(root->left, key);
    else root->right = insert(root->right, key);
    return root;
}

// Function to find the minimum node in a BST
TreeNode* findMin(TreeNode* root) {
    while (root->left) root = root->left;
    return root;
}

// Function to delete a node from BST
TreeNode* deleteNode(TreeNode* root, int key) {
    if (!root) return root;

    if (key < root->key) {
        root->left = deleteNode(root->left, key);
    } else if (key > root->key) {
        root->right = deleteNode(root->right, key);
    } else {
        // Node with only one child or no child
        if (!root->left) {
            TreeNode* temp = root->right;
            free(root);
            return temp;
        } else if (!root->right) {
            TreeNode* temp = root->left;
            free(root);
            return temp;
        }
        // Node with two children, find the inorder successor
        TreeNode* temp = findMin(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key);
    }
    return root;
}

// Function to print BST in in-order traversal
void inorderTraversal(TreeNode* root) {
    if (!root) return;
    inorderTraversal(root->left);
    printf("%d ", root->key);
    inorderTraversal(root->right);
}

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

    TreeNode* root = NULL;

    // Read the BST node values and insert into BST
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        root = insert(root, key);
    }

    scanf("%d", &V);

    // Check if the node exists before deletion
    TreeNode* temp = root;
    int found = 0;

    while (temp) {
        if (temp->key == V) {
            found = 1;
            break;
        }
        temp = (V < temp->key) ? temp->left : temp->right;
    }

    if (found) {
        root = deleteNode(root, V);
    }

    // Print in-order traversal of BST
    inorderTraversal(root);

    return 0;
}
int main()
{
    int N, rootValue, V;
    scanf("%d", &N);
    struct TreeNode* root = NULL;
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }
    scanf("%d", &V);
    root = deleteNode(root, V);
    inorderTraversal(root);
    return 0;
}
```

Status : Wrong ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Mike is learning about Binary Search Trees (BSTs) and wants to implement various operations on them. He wants to write a basic program for creating a BST, inserting nodes, and printing the tree in the pre-order traversal.

Write a program to help him solve this program.

Input Format

The first line of input consists of an integer N, representing the number of values to insert into the BST.

The second line consists of N space-separated integers, representing the values to insert into the BST.

Output Format

The output prints the space-separated values of the BST in the pre-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 1 5 2 4
Output: 3 1 2 5 4

Answer

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

struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

struct Node* createNode(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->left = newNode->right = NULL;
   return newNode;
}
// You are using GCC
struct Node* insert(struct Node* root, int value) {
   //Type your code here
   if(root==NULL)
   {
   return createNode(value);
   }
   else if(value<root->data)
   {
   root->left=insert(root->left,value);
   }
   else if(value>root->data)
```

```
    {
    root->right=insert(root->right,value);
    }
    return root;
}

void printPreorder(struct Node* node) {
    //Type your code here
    if(node!=NULL)
    {
        printf("%d",node->data);
        printPreorder(node->left);
        printPreorder(node->right);
    }
}
int main() {
    struct Node* root = NULL;

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

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

    printPreorder(root);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

Input Format

The first line of input consists of an integer n, representing the number of nodes in the binary search tree.

The second line consists of the values of the nodes, separated by space as integers.

The third line consists of an integer representing, the value that is to be searched.

Output Format

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 3 10 1 6 14 23
6
Output: Value 6 is found in the tree.

Answer

```
// You are using GCC

struct Node* insertNode(struct Node* root, int value) {
   //Type your code here
   struct Node* nn=(struct Node*)malloc(sizeof(struct Node));
   if(root==NULL)
   {
   return createNode(value);
   }
   else if(value<root->data);
   {
   root->left=insertNode(root->left,value);
   }
   else if(value>root->data);
   {
   root->right=insertNode(root->right,value);
   }
   return root;
}
```

```
struct Node* searchNode(struct Node* root, int value) {
  //Type your code here
  if(root==NULL)
  {
    return NULL;
  }
  else if(value<root->data)
  {
    return searchNode(root->left,value);
  }
  else if(value>root->data)
  {
    return searchNode(root->right,value);
  }
  return root;
}
```

Status : Wrong ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John, a computer science student, is learning about binary search trees (BST) and their properties. He decides to write a program to create a BST, display it in post-order traversal, and find the minimum value present in the tree.

Help him by implementing the program.

Input Format

The first line of input consists of an integer N, representing the number of elements to insert into the BST.

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

Output Format

The first line of output prints the space-separated elements of the BST in post-order traversal.

The second line prints the minimum value found in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 10 15
Output: 15 10 5
The minimum value in the BST is: 5

Answer

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// You are using GCC
struct Node* insert(struct Node* root, int data) {
    //Type your code here
    if(root==NULL)
    return createNode(data);
    else if(data<root->data)
    root->left=insert(root->left,data);
    else if(data>root->data)
    root->right=insert(root->right,data);
    return root;
}

void displayTreePostOrder(struct Node* root) {
    //Type your code here
    if(root!=NULL)
    {
        displayTreePostOrder(root->left);
        displayTreePostOrder(root->right);
        printf("%d",root->data);
    }
}

int findMinValue(struct Node* root) {
    //Type your code here
    if(root==NULL)
    return 0;
    else if(root->left==NULL)
    return root->data;
    else
    return findMinValue(root->left);
}

int main() {
    struct Node* root = NULL;
    int n, data;
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        root = insert(root, data);
    }

    displayTreePostOrder(root);
    printf("\n");

    int minValue = findMinValue(root);
    printf("The minimum value in the BST is: %d", minValue);

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Miruthulaa Manickam
Email: 240701310@rajalakshmi.edu.in
Roll no: 240701310
Phone: 7900440030
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In his computer science class, John is learning about Binary Search Trees (BST). He wants to build a BST and find the maximum value in the tree.

Help him by writing a program to insert nodes into a BST and find the maximum value in the tree.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the nodes to insert into the BST.

Output Format

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7

Output: 15

Answer

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

struct TreeNode {
   int data;
   struct TreeNode* left;
   struct TreeNode* right;
};
struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
   newNode->data = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}
// You are using GCC
struct TreeNode* insert(struct TreeNode* root, int key) {
   //Type your code here
   if(root==NULL)
   return createNode(key);
   else if(key<root->data)
   root->left=insert(root->left,key);
   else if(key>root->data)
   root->right=insert(root->right,key);
   return root;
}

int findMax(struct TreeNode* root) {
   //Type your code here
   if(root==NULL)
   return 0;
   else if(root->right==NULL)
   return root->data;
   else
   return findMax(root->right);
}

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

   struct TreeNode* root = NULL;

   for (int i = 0; i < N; i++) {
       int key;
       scanf("%d", &key);
       if (i == 0) rootValue = key;
       root = insert(root, key);
   }

   int maxVal = findMax(root);
   if (maxVal != -1) {
       printf("%d", maxVal);
   }

   return 0;
}
```

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