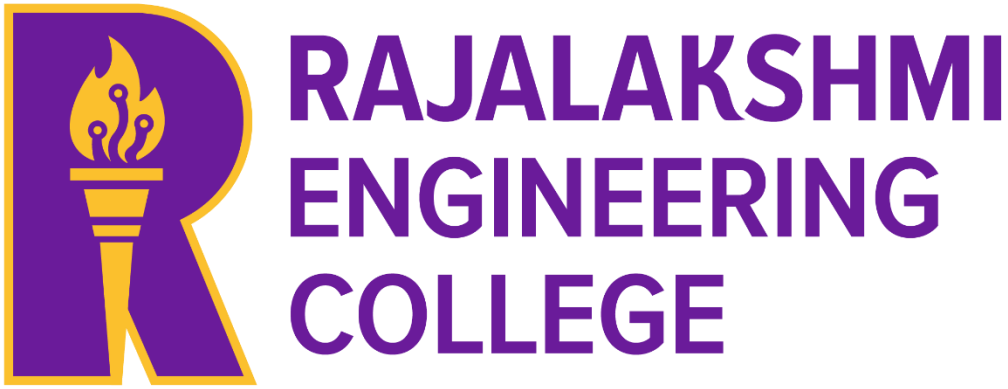
RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

RAJALAKSHMI NAGAR, THANDALAM- 602 105



CS23231 - DATA STRUCTURES

LABORATORY RECORD NOTEBOOK

NAME: ……

YEAR/SEMESTER: ……

BRANCH/SECTION: ……

REGISTER NO: ……

COLLEGE ROLL NO: ……

ACADEMIC YEAR: 20….. -20…..

RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

RAJALAKSHMI NAGAR, THANDALAM- 602 105

BONAFIDE CERTIFICATE

NAME: …… **BRANCH/SECTION:** ……

ACADEMIC YEAR: 20….. -20….. **SEMESTER:** ……..

REGISTER NO:

Certified that this is a Bonafide record of work done by the above student in the **CS23231 - DATA STRUCTURES** during the year 20 - 20

Signature of Faculty In-charge

Submitted for the Practical Examination Held on: ……..

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR CODE	PAGE NO	STAFF SIGN
1		Implementation of Single LinkedList (Insertion, Deletion, and Display).			
2		Implementation of Doubly LinkedList (Insertion, Deletion, and Display).			
3		Implementation of a Stack using an Array and a linked implementation.			
4		Implementation of a Queue using an Array and a linked implementation.			
5		Implementation of Binary Search Tree and perform Tree Traversal Techniques.			
6		Program to perform Sorting (Quick sort, Merge sort)			
7		Program to perform Hashing (Linear Probing, Rehashing)			

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 : 5

Section 1 : Coding

1. 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

```
// You are using GCC
#include<stdio.h>
int main()
{
    int n,max=0,low=0;
    float m;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d\n",&arr[i]);
        printf("Employee %d: %d\n",i+1,arr[i]);
        m+=arr[i];
        if(arr[i]>max)
        {
            max=arr[i];
        }
    }
    low=arr[0];
    for(int i=1;i<n;i++)
    {
        if(arr[i]<low)
        {
            low=arr[i];
        }
    }
    printf("Average Salary: %.2f\n",m/n);
    printf("Highest Salary: %d\n",max);
    printf("Lowest Salary: %d\n",low);
}
```

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

2. 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

```
// You are using GCC
#include<stdio.h>
void smallest(int a[],int n){
    int r=a[0];
    for(int i=1;i<n;i++)
    {
        if(a[i]<r)
        {
            r=a[i];
        }
    }
    printf("The smallest number is: %d",r);
}
void largest(int a[],int n){
    int h=a[0];
```

```
    for(int i=1;i<n;i++)
    {
        if(a[i]>h)
        {
            h=a[i];
        }
    }
    printf("The largest number is: %d",h);
}
void sum(int a[],int n){
    int s=0;
    for(int i=0;i<n;i++)
    {
        s+=a[i];
    }
    printf("The sum of the numbers is: %d",s);
}
void average(int a[],int n)
{
    float r=0;
    for(int i=0;i<n;i++)
    {
        r+=a[i];
    }
    printf("The average of the numbers is: %.2f",r/n);
}
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    int m;
    do{
    scanf("%d",&m);
    switch(m){
    case 1:
     smallest(a,n);
      break;
```

```
    case 2:
      largest(a,n);
      break;
    case 3:
      sum(a,n);
      break;
    case 4:
      average(a,n);
      break;
    case 5:
      printf("Exiting the program");
      break;
    default:
      printf("Invalid choice! Please enter a valid option (1-5).");
      break;
    }
    }while(m!=5);
    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) {
    //Type your code here
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<=i;j++)
        {
            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

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) {
    //Type your code here
    for(int i=0;i<rows;i++)
    {
    int sum=0;
    for(int j=0;j<cols;j++)
    {
        sum+=matrix[i][j];
    }
    printf("%d ",sum);
    }
}

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***

5. 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

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

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 2
Total Mark : 10
Marks Obtained : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int coeff;
    int expo;
    struct node *next;
};
typedef struct  node Node;
Node *create(int coeff,int expo)
{
    Node *newnode;
    newnode=(Node *)malloc(sizeof(Node));
    newnode->coeff=coeff;
    newnode->expo=expo;
    newnode->next=NULL;
    return newnode;
}

void insert(Node **head,int coeff,int expo){
```

```
    Node *newnode=create(coeff,expo);
    newnode->next=*head;
    *head=newnode;
    return;
}
int sumpol(Node *head)
{
    int sum=0;
    while(head!=NULL){
        sum+=head->coeff;
        head=head->next;
    }
    return sum;
}

int main()
{
    Node *poly1=NULL,*poly2=NULL;
    int n,m;
    scanf("%d",&n);
    int coeff ,expo;
    for(int i=0;i<n;i++){
        scanf("%d %d",&coeff,&expo);
        insert(&poly1,coeff,expo);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++)
    {
        scanf("%d %d",&coeff,&expo);
        insert(&poly2,coeff,expo);
    }
    int totalsum=sumpol(poly1)+sumpol(poly2);
    printf("%d",totalsum);
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;

// You are using GCC
typedef struct node Node;

void insert(int x)
{
    Node *newnode;
    newnode=(Node *)malloc(sizeof(Node));
    newnode->data=x;
    newnode->next=NULL;
    if(head==NULL){
```

```
        head=newnode;
        tail=newnode;
    }
    else{
        tail->next=newnode;
        tail=newnode;
    }
}
void deleteNode(int pos)
{
    if(head==NULL){
        tail=NULL;
        printf("Invalid position. Deletion not possible.");
        return;
    }
    if(pos<1){
        printf("Invalid position. Deletion not possible.");
        return;
    }
    struct node *temp=NULL;
    if(pos==1){
        temp=head;
        head=head->next;
        free(temp);
        temp=NULL;
        display_List();
        return;
    }
    int count=1;
    struct node *current=head;

    while(current!=NULL && count<pos)
    {
        temp=current;
        current=current->next;
        count++;
    }
    if(current==NULL)
    {
        printf("Invalid position. Deletion not possible.");
        return;
    }
```

```
    else if(current!=NULL){
        temp->next=current->next;
        free(current);
        current=NULL;

        if(temp->next==NULL){
            tail=temp;
        }
        display_List();

        return;
    }
}
void display_List(){
    struct node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;
    }
    return;
}


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: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
};

// You are using GCC
typedef struct Node node;
void insertAtFront(node** head,int x)
{
    node *newnode;
    newnode=(node *)malloc(sizeof(node));
    newnode->data=x;
    newnode->next=*head;
    *head=newnode;
}

void printList(node *head){
    Node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;
    }
}
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: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

```
// You are using GCC

#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node *next;
};
typedef struct node Node;
void insert(Node **head,int x)
{
   Node *newnode;
   newnode=(Node *)malloc(sizeof(Node));
   newnode->data=x;
   newnode->next=NULL;
   if(*head==NULL)
   {
      *head=newnode;
      return;
   }
   Node *current=*head;
   while(current->next!=NULL)
   {
      current=current->next;
   }
   current->next=newnode;
   return;
}
```

```
void display(Node *head)
{
    Node *current=head;
    while(current!=NULL)
    {
        printf("%d ",current->data);
        current=current->next;
    }
    return;
}
int main()
{
    Node *head=NULL;
    int n;
    scanf("%d",&n);
    int a;
    for(int i=0;i<n;i++){
        scanf("%d",&a);
        insert(&head,a);
    }
    display(head);
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Aarav is working on a program to analyze his test scores, which are stored in a doubly linked list. He needs a solution to input scores into the list and determine the highest score.

Help him by providing code that lets users enter test scores into the doubly linked list and find the maximum score efficiently.

Input Format

The first line consists of an integer N, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of N space-separated integers, denoting the score to be inserted.

Output Format

The output prints an integer, representing the highest score present in the list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
89 71 2 70

Output: 89

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


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


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


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

```
    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;

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


int findMax(Node* head) {
    if (head == NULL)
        return -1;

    int max = head->data;
    Node* temp = head->next;

    while (temp != NULL) {
        if (temp->data > max)
            max = temp->data;
        temp = temp->next;
    }
    return max;
}

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

    Node* head = NULL;
    int score;

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

    int maxScore = findMax(head);
    printf("%d\n", maxScore);


    Node* temp;
```

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

    return 0;
}
```

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

2. Problem Statement

Vanessa is learning about the doubly linked list data structure and is eager to play around with it. She decides to find out how the elements are inserted at the beginning and end of the list.

Help her implement a program for the same.

Input Format

The first line of input contains an integer N, representing the size of the doubly linked list.

The next line contains N space-separated integers, each representing the values to be inserted into the doubly linked list.

Output Format

The first line of output prints the integers, after inserting them at the beginning, separated by space.

The second line prints the integers, after inserting at the end, separated by space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

1 2 3 4 5

Output: 5 4 3 2 1

1 2 3 4 5

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


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


void insertAtBeginning(struct Node** head_ref, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = *head_ref;

    if (*head_ref != NULL)
        (*head_ref)->prev = newNode;

    *head_ref = newNode;
}


void insertAtEnd(struct Node** head_ref, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    struct Node* temp = *head_ref;
    newNode->data = data;
    newNode->next = NULL;

    if (*head_ref == NULL) {
        newNode->prev = NULL;
        *head_ref = newNode;
        return;
    }
```

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

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


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

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

    struct Node* head_beginning = NULL;
    struct Node* head_end = NULL;

    for (i = 0; i < N; i++) {
        scanf("%d", &value);
        insertAtBeginning(&head_beginning, value);
        insertAtEnd(&head_end, value);
    }

    printList(head_beginning);
    printList(head_end);

    return 0;
}
```

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

3. Problem Statement

You are required to implement a program that deals with a doubly linked

list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list.Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list.Display the List: Display the elements of the doubly linked list.

Input Format

The first line of input consists of an integer n, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of n space-separated integers, denoting the elements to be inserted at the end.

The third line consists of integer m, representing the new element to be inserted.

The fourth line consists of an integer p, representing the position at which the new element should be inserted (1-based indexing).

Output Format

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints the original list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 25 34 48 57
35
4

Output: 10 25 34 35 48 57

Answer

// You are using GCC

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


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


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


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


int insertAtPosition(Node** head, int data, int position) {
    if (position < 1) return 0;

    Node* newNode = createNode(data);

    if (position == 1) {
        newNode->next = *head;
        if (*head != NULL)
```

```
        (*head)->prev = newNode;
        *head = newNode;
        return 1;
    }

    Node* temp = *head;
    for (int i = 1; i < position - 1; i++) {
        if (temp == NULL)
            return 0;
        temp = temp->next;
    }

    if (temp == NULL)
        return 0;

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

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

    temp->next = newNode;
    return 1;
}


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


Node* copyList(Node* head) {
    if (head == NULL) return NULL;
    Node* newHead = NULL, *tail = NULL;
    while (head != NULL) {
        Node* newNode = createNode(head->data);
        if (newHead == NULL) {
```

```
            newHead = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
        head = head->next;
    }
    return newHead;
}


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

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

    Node* originalList = copyList(head);

    int m, p;
    scanf("%d", &m);
    scanf("%d", &p);

    if (insertAtPosition(&head, m, p)) {
        displayList(head);
    } else {
        printf("Invalid position\n");
        displayList(originalList);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 27.5

Section 1 : Coding

1. Problem Statement

Hasini is studying polynomials in her class. Her teacher has introduced a new concept of two polynomials using linked lists.

The teacher provides Hasini with a program that takes two polynomials as input, represented as linked lists, and then displays them together. The polynomials are simplified and should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

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 first line of output prints the first polynomial.

The second line of output prints the second polynomial.

The polynomials should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2
2 1
3 0
3
2 2
1 1
4 0
Output: 1x^2 + 2x + 3
2x^2 + 1x + 4

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
void printTerm(int coeff, int exp,int isFirstTerm)
{
    if(coeff==0)
    return;
    if(coeff>0 && !isFirstTerm)
    printf("+");
```

```
    else if(coeff<0)
    printf("-");
    if(abs(coeff)!=1||exp==0)
    printf("%d",abs(coeff));
    else if(coeff == 1 && exp!=0)
    printf("1");
    else if(coeff==-1 && exp!=0)
    printf( "1");
    if(exp>0)
    {
       printf("x");
       if(exp>1)
       printf("^%d",exp);
    }
}
void printPolynomial(int *coeffs, int *exps,int numTerms)
{
    int isFirstTerm=1;
    for(int i=0;i<numTerms;i++)
    {
       if(coeffs[i]!=0){
          printTerm(coeffs[i],exps[i],isFirstTerm);
          isFirstTerm=0;
       }
    }
    if(isFirstTerm)
      printf("0");
    printf("\n");
}
void sortPolynomial(int *coeffs,int *exps,int numTerms)
{
    for(int i=0;i<numTerms;i++)
    {
       for(int j=i+1;j<numTerms;j++)
       {
          if(exps[i]<exps[j])
          {
             int tempCoeff=coeffs[i];
             coeffs[i]=coeffs[j];
             coeffs[j]=tempCoeff;

             int tempExp=exps[i];
```

```
            exps[i]=exps[j];
            exps[j]=tempExp;
        }
    }
  }
}
int main()
{
  int n,m,coeff,exp;
  scanf("%d",&n);
  int *coeffs1=(int* )malloc(n* sizeof(int));
  int *exps1=(int* )malloc(n* sizeof(int));
  if(!coeffs1 || !exps1)
  {
    printf("Memory allocation failed.\n");
    return 1;
  }
  for(int i=0;i<n;i++)
  {
    scanf("%d %d",&coeff,&exp);
    coeffs1[i]=coeff;
    exps1[i]=exp;
  }
  sortPolynomial(coeffs1,exps1,n);

  scanf("%d",&m);
  int *coeffs2=(int* )malloc(m* sizeof(int));
  int *exps2=(int *)malloc(m* sizeof(int));
  if(!coeffs2 || !exps2)
  {
    printf("Memory allocation failed.\n");
    return 1;
  }
  for(int i=0;i<m;i++)
  {
    scanf("%d %d",&coeff,&exp);
    coeffs2[i]=coeff;
    exps2[i]=exp;
  }
  sortPolynomial(coeffs2,exps2,m);
  printPolynomial(coeffs1,exps1,n);
  printPolynomial(coeffs2,exps2,m);
```

```
    free(coeffs1);
    free(exps1);
    free(coeffs2);
    free(exps2);
    return 0;
}
```

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

2. Problem Statement

Akila is a tech enthusiast and wants to write a program to add two polynomials. Each polynomial is represented as a linked list, where each node in the list represents a term in the polynomial.

A term in the polynomial is represented in the format ax^b, where a is the coefficient and b is the exponent.

Akila needs your help to implement a program that takes two polynomials as input, adds them, and stores the result in ascending order in a new polynomial-linked list. Write a program to help her.

Input Format

The input consists of lines containing pairs of integers representing the coefficients and exponents of polynomial terms.

Each line represents a single term, with the coefficient and exponent separated by a space.

The input for each polynomial ends with a line containing "0 0".

Output Format

The output consists of three lines representing the first, second, and resulting polynomial after the addition operation, with terms sorted in ascending order of exponents.

Each line contains terms of the polynomial in the format "coefficientx^exponent", separated by " + ".

If the resulting polynomial is zero, the output is "0".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 4
2 3
1 2
0 0
1 2
2 3
3 4
0 0
Output: 1x^2 + 2x^3 + 3x^4
1x^2 + 2x^3 + 3x^4
2x^2 + 4x^3 + 6x^4

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int coeff;
    int expo;
    struct Node* next;
} Node;

// Create a new node
Node* createNode(int coeff, int expo) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->expo = expo;
    newNode->next = NULL;
    return newNode;
}

// Insert a term into the polynomial in sorted order
```

```
Node* insertTerm(Node* head, int coeff, int expo) {
    if (coeff == 0) return head;

    Node* newNode = createNode(coeff, expo);
    if (head == NULL || expo < head->expo) {
        newNode->next = head;
        return newNode;
    }

    Node *curr = head, *prev = NULL;
    while (curr && curr->expo < expo) {
        prev = curr;
        curr = curr->next;
    }

    if (curr && curr->expo == expo) {
        curr->coeff += coeff;
        if (curr->coeff == 0) {
            if (prev) prev->next = curr->next;
            else head = curr->next;
            free(curr);
        }
        free(newNode);
    } else {
        newNode->next = curr;
        if (prev) prev->next = newNode;
        else head = newNode;
    }
    return head;
}

// Build a polynomial from input
Node* buildPolynomial() {
    Node* head = NULL;
    int coeff, expo;
    while (1) {
        scanf("%d %d", &coeff, &expo);
        if (coeff == 0 && expo == 0)
            break;
        head = insertTerm(head, coeff, expo);
    }
    return head;
```

```
}

// Add two polynomials
Node* addPolynomials(Node* poly1, Node* poly2) {
    Node* result = NULL;
    while (poly1) {
        result = insertTerm(result, poly1->coeff, poly1->expo);
        poly1 = poly1->next;
    }
    while (poly2) {
        result = insertTerm(result, poly2->coeff, poly2->expo);
        poly2 = poly2->next;
    }
    return result;
}

// Print the polynomial
void printPolynomial(Node* head) {
    if (!head) {
        printf("0\n");
        return;
    }
    while (head) {
        printf("%dx^%d", head->coeff, head->expo);
        if (head->next)
            printf(" + ");
        head = head->next;
    }
    printf("\n");
}

// Free the list
void freeList(Node* head) {
    while (head) {
        Node* temp = head;
        head = head->next;
        free(temp);
    }
}

// Main
int main() {
```

```
    Node *poly1, *poly2, *result;

    poly1 = buildPolynomial();
    poly2 = buildPolynomial();
    result = addPolynomials(poly1, poly2);

    printPolynomial(poly1);
    printPolynomial(poly2);
    printPolynomial(result);

    freeList(poly1);
    freeList(poly2);
    freeList(result);

    return 0;
}
```

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

3. Problem Statement

Keerthi is a tech enthusiast and is fascinated by polynomial expressions. She loves to perform various operations on polynomials.

Today, she is working on a program to multiply two polynomials and delete a specific term from the result.

Keerthi needs your help to implement this program. She wants to take the coefficients and exponents of the terms of the two polynomials as input, perform the multiplication, and then allow the user to specify an exponent for deletion from the resulting polynomial, and display the result.

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 representing the coefficient and the exponent of the term in the first polynomial.

The next line 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 representing the coefficient and the exponent of the term in the second polynomial.

The last line consists of an integer, representing the exponent of the term that Keerthi wants to delete from the multiplied polynomial.

Output Format

The first line of output displays the resulting polynomial after multiplication.

The second line displays the resulting polynomial after deleting the specified term.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
2 2
3 1
4 0
2
1 2
2 1
2
Output: Result of the multiplication: 2x^4 + 7x^3 + 10x^2 + 8x
Result after deleting the term: 2x^4 + 7x^3 + 8x

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

#define MAX_TERMS 100

// Structure to represent a term of the polynomial
typedef struct {
    int coeff;
    int exp;
```

```
} Term;

// Function to multiply two polynomials
void multiplyPolynomials(Term poly1[], int n, Term poly2[], int m, int result[]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            int exp_sum = poly1[i].exp + poly2[j].exp;
            result[exp_sum] += poly1[i].coeff * poly2[j].coeff;
        }
    }
}

// Function to print the polynomial from the result array
void printPolynomial(int result[], int max_exp, int skip_exp) {
    int first = 1;
    for (int i = max_exp; i >= 0; i--) {
        if (i == skip_exp || result[i] == 0)
            continue;

        if (!first) printf(" + ");
        if (i == 0)
            printf("%d", result[i]);
        else if (i == 1)
            printf("%dx", result[i]);
        else
            printf("%dx^%d", result[i], i);
        first = 0;
    }
    if (first) printf("0"); // If all terms were deleted
    printf("\n");
}

int main() {
    int n, m, delete_exp;
    Term poly1[5], poly2[5];
    int result[MAX_TERMS] = {0};

    // Read first polynomial
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &poly1[i].coeff, &poly1[i].exp);
    }
```

```
    // Read second polynomial
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &poly2[i].coeff, &poly2[i].exp);
    }

    // Read exponent to delete
    scanf("%d", &delete_exp);

    // Multiply
    multiplyPolynomials(poly1, n, poly2, m, result);

    // Find maximum exponent for output loop
    int max_exp = 0;
    for (int i = 0; i < MAX_TERMS; i++) {
        if (result[i] != 0 && i > max_exp) {
            max_exp = i;
        }
    }

    // Output
    printf("Result of the multiplication: ");
    printPolynomial(result, max_exp, -1);

    printf("Result after deleting the term: ");
    printPolynomial(result, max_exp, delete_exp);

    return 0;
}
```

Status : Partially correct **Marks : 7.5/10**

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_PAH_modified

Attempt : 1
Total Mark : 5
Marks Obtained : 3.8

Section 1 : Coding

1. Problem Statement

Emily is developing a program to manage a singly linked list. The program should allow users to perform various operations on the linked list, such as inserting elements at the beginning or end, deleting elements from the beginning or end, inserting before or after a specific value, and deleting elements before or after a specific value. After each operation, the updated linked list should be displayed.

Your task is to help Emily in implementing the same.

Input Format

The first line contains an integer choice, representing the operation to perform:

- For choice 1 to create the linked list. The next lines contain space-separated

integers, with -1 indicating the end of input.
- For choice 2 to display the linked list.
- For choice 3 to insert a node at the beginning. The next line contains an integer data representing the value to insert.
- For choice 4 to insert a node at the end. The next line contains an integer data representing the value to insert.
- For choice 5 to insert a node before a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 6 to insert a node after a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 7 to delete a node from the beginning.
- For choice 8 to delete a node from the end.
- For choice 9 to delete a node before a specific value. The next line contains an integer value representing the node before which deletion occurs.
- For choice 10 to delete a node after a specific value. The next line contains an integer value representing the node after which deletion occurs.
- For choice 11 to exit the program.

Output Format

For choice 1, print "LINKED LIST CREATED".

For choice 2, print the linked list as space-separated integers on a single line. If the list is empty, print "The list is empty".

For choice 3, 4, 5, and 6, print the updated linked list with a message indicating the insertion operation.

For choice 7, 8, 9, and 10, print the updated linked list with a message indicating the deletion operation.

For any operation that is not possible print an appropriate error message such as "Value not found in the list".

For choice 11 terminate the program.

For any invalid option, print "Invalid option! Please try again".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
5
3
7
-1
2
11

Output: LINKED LIST CREATED
5 3 7

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

Node* head = NULL;

// Function prototypes
void createList();
void displayList();
void insertAtBeginning(int data);
void insertAtEnd(int data);
void insertBeforeValue(int value, int data);
void insertAfterValue(int value, int data);
void deleteFromBeginning();
void deleteFromEnd();
void deleteBeforeValue(int value);
void deleteAfterValue(int value);

// Create List
void createList() {
    int data;
    head = NULL;
    while (1) {
        scanf("%d", &data);
```

```
        if (data == -1) break;
        insertAtEnd(data);
    }
    printf("LINKED LIST CREATED\n");
}

// Display List
void displayList() {
    if (!head) {
        printf("The list is empty\n");
        return;
    }
    Node* temp = head;
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Insert at Beginning
void insertAtBeginning(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("The linked list after insertion at the beginning is: ");
    displayList();
}

// Insert at End
void insertAtEnd(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;

    if (!head) {
        head = newNode;
        return;
    }

    Node* temp = head;
```

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

// Insert Before Value
void insertBeforeValue(int value, int data) {
    if (!head) {
        printf("Value not found in the list\n");
        return;
    }

    if (head->data == value) {
        insertAtBeginning(data);
        return;
    }

    Node* temp = head;
    Node* prev = NULL;

    while (temp && temp->data != value) {
        prev = temp;
        temp = temp->next;
    }

    if (!temp) {
        printf("Value not found in the list\n");
        return;
    }

    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = temp;
    prev->next = newNode;

    printf("The linked list after insertion before a value is: ");
    displayList();
}

// Insert After Value
void insertAfterValue(int value, int data) {
    Node* temp = head;
```

```
    while (temp && temp->data != value)
        temp = temp->next;

    if (!temp) {
        printf("Value not found in the list\n");
        return;
    }

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

    printf("The linked list after insertion after a value is: ");
    displayList();
}

// Delete from Beginning
void deleteFromBeginning() {
    if (!head) {
        printf("The list is empty\n");
        return;
    }

    Node* temp = head;
    head = head->next;
    free(temp);

    printf("The linked list after deletion from the beginning is: ");
    displayList();
}

// Delete from End
void deleteFromEnd() {
    if (!head) {
        printf("The list is empty\n");
        return;
    }

    if (!head->next) {
        free(head);
        head = NULL;
```

```
        printf("The linked list after deletion from the end is: ");
        displayList();
        return;
    }

    Node* temp = head;
    Node* prev = NULL;

    while (temp->next) {
        prev = temp;
        temp = temp->next;
    }

    prev->next = NULL;
    free(temp);

    printf("The linked list after deletion from the end is: ");
    displayList();
}

// Delete Before Value
void deleteBeforeValue(int value) {
    if (!head || !head->next) {
        printf("Value not found in the list\n");
        return;
    }

    if (head->data == value) {
        printf("Value not found in the list\n");
        return;
    }

    Node* temp = head;
    Node* prev = NULL;
    Node* prePrev = NULL;

    while (temp->next && temp->next->data != value) {
        prePrev = prev;
        prev = temp;
        temp = temp->next;
    }
```

```
    if (!temp->next) {
        printf("Value not found in the list\n");
        return;
    }

    if (!prev) {
        head = temp->next;
        free(temp);
    } else {
        prePrev->next = temp->next;
        free(temp);
    }

    printf("The linked list after deletion before a value is: ");
    displayList();
}

// Delete After Value
void deleteAfterValue(int value) {
    Node* temp = head;
    while (temp && temp->data != value)
        temp = temp->next;

    if (!temp || !temp->next) {
        printf("Value not found in the list\n");
        return;
    }

    Node* delNode = temp->next;
    temp->next = delNode->next;
    free(delNode);

    printf("The linked list after deletion after a value is: ");
    displayList();
}

// Main
int main() {
    int choice, data, value;
    while (1) {
        scanf("%d", &choice);
        switch (choice) {
```

```
case 1:
    createList();
    break;
case 2:
    displayList();
    break;
case 3:
    scanf("%d", &data);
    insertAtBeginning(data);
    break;
case 4:
    scanf("%d", &data);
    insertAtEnd(data);
    printf("The linked list after insertion at the end is: ");
    displayList();
    break;
case 5:
    scanf("%d %d", &value, &data);
    insertBeforeValue(value, data);
    break;
case 6:
    scanf("%d %d", &value, &data);
    insertAfterValue(value, data);
    break;
case 7:
    deleteFromBeginning();
    break;
case 8:
    deleteFromEnd();
    break;
case 9:
    scanf("%d", &value);
    deleteBeforeValue(value);
    break;
case 10:
    scanf("%d", &value);
    deleteAfterValue(value);
    break;
case 11:
    return 0;
default:
    printf("Invalid option! Please try again\n");
```

```
    }
  }
}
```

Status : Partially correct **Marks : 0.4/1**

2. Problem Statement

Write a program to manage a singly linked list. The program should allow users to perform various operations on the linked list, such as inserting elements at the beginning or end, deleting elements from the beginning or end, inserting before or after a specific value, and deleting elements before or after a specific value. After each operation, the updated linked list should be displayed.

Input Format

The first line contains an integer choice, representing the operation to perform:

- For choice 1 to create the linked list. The next lines contain space-separated integers, with -1 indicating the end of input.
- For choice 2 to display the linked list.
- For choice 3 to insert a node at the beginning. The next line contains an integer data representing the value to insert.
- For choice 4 to insert a node at the end. The next line contains an integer data representing the value to insert.
- For choice 5 to insert a node before a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 6 to insert a node after a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 7 to delete a node from the beginning.
- For choice 8 to delete a node from the end.
- For choice 9 to delete a node before a specific value. The next line contains an integer value representing the node before which deletion occurs.
- For choice 10 to delete a node after a specific value. The next line contains an integer value representing the node after which deletion occurs.
- For choice 11 to exit the program.

Output Format

For choice 1, print "LINKED LIST CREATED".

For choice 2, print the linked list as space-separated integers on a single line. If the list is empty, print "The list is empty".

For choice 3, 4, 5, and 6, print the updated linked list with a message indicating the insertion operation.

For choice 7, 8, 9, and 10, print the updated linked list with a message indicating the deletion operation.

For any operation that is not possible print an appropriate error message such as "Value not found in the list".

For choice 11 terminate the program.

For any invalid option, print "Invalid option! Please try again".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
5
3
7
-1
2
11
Output: LINKED LIST CREATED
5 3 7

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

```
Node* head = NULL;

// Create list
void createList() {
    int value;
    head = NULL;
    while (scanf("%d", &value), value != -1) {
        Node* newNode = (Node*)malloc(sizeof(Node));
        newNode->data = value;
        newNode->next = NULL;
        if (!head) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next)
                temp = temp->next;
            temp->next = newNode;
        }
    }
    printf("LINKED LIST CREATED\n");
}

// Display list
void displayList() {
    if (!head) {
        printf("The list is empty\n");
        return;
    }
    Node* temp = head;
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Insert at beginning
void insertAtBeginning(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
```

```
    printf("The linked list after insertion at the beginning is: ");
    displayList();
}

// Insert at end
void insertAtEnd(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;
    if (!head) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next)
            temp = temp->next;
        temp->next = newNode;
    }
    printf("The linked list after insertion at the end is: ");
    displayList();
}

// Insert before value
void insertBefore(int target, int value) {
    if (!head) {
        printf("Value not found in the list\n");
        return;
    }

    if (head->data == target) {
        insertAtBeginning(value);
        return;
    }

    Node* prev = NULL, * curr = head;
    while (curr && curr->data != target) {
        prev = curr;
        curr = curr->next;
    }

    if (!curr) {
        printf("Value not found in the list\n");
        return;
```

```
    }

    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = curr;
    prev->next = newNode;

    printf("The linked list after insertion before a value is: ");
    displayList();
}

// Insert after value
void insertAfter(int target, int value) {
    Node* curr = head;
    while (curr && curr->data != target)
        curr = curr->next;

    if (!curr) {
        printf("Value not found in the list\n");
        return;
    }

    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = curr->next;
    curr->next = newNode;

    printf("The linked list after insertion after a value is: ");
    displayList();
}

// Delete from beginning
void deleteFromBeginning() {
    if (!head) {
        printf("The list is empty\n");
        return;
    }

    Node* temp = head;
    head = head->next;
    free(temp);
```

```
    printf("The linked list after deletion from the beginning is: ");
    displayList();
}

// Delete from end
void deleteFromEnd() {
    if (!head) {
        printf("The list is empty\n");
        return;
    }

    if (!head->next) {
        free(head);
        head = NULL;
        printf("The linked list after deletion from the end is: ");
        displayList();
        return;
    }

    Node* curr = head, * prev = NULL;
    while (curr->next) {
        prev = curr;
        curr = curr->next;
    }

    prev->next = NULL;
    free(curr);

    printf("The linked list after deletion from the end is: ");
    displayList();
}

// Delete before value
void deleteBefore(int target) {
    if (!head || head->data == target) {
        printf("Value not found in the list\n");
        return;
    }

    Node *prevPrev = NULL, *prev = head, *curr = head->next;
    while (curr && curr->data != target) {
        prevPrev = prev;
```

```
        prev = curr;
        curr = curr->next;
    }

    if (!curr) {
        printf("Value not found in the list\n");
        return;
    }

    if (!prevPrev) {
        head = curr;
    } else {
        prevPrev->next = curr;
    }
    free(prev);

    printf("The linked list after deletion before a value is: ");
    displayList();
}

// Delete after value
void deleteAfter(int target) {
    Node* curr = head;
    while (curr && curr->data != target)
        curr = curr->next;

    if (!curr || !curr->next) {
        printf("Value not found in the list\n");
        return;
    }

    Node* temp = curr->next;
    curr->next = temp->next;
    free(temp);

    printf("The linked list after deletion after a value is: ");
    displayList();
}

// Main
int main() {
    int choice, value, newValue;
```

```
while (1) {
    if (scanf("%d", &choice) != 1) break;

    switch (choice) {
        case 1:
            createList();
            break;
        case 2:
            displayList();
            break;
        case 3:
            scanf("%d", &value);
            insertAtBeginning(value);
            break;
        case 4:
            scanf("%d", &value);
            insertAtEnd(value);
            break;
        case 5:
            scanf("%d %d", &value, &newValue);
            insertBefore(value, newValue);
            break;
        case 6:
            scanf("%d %d", &value, &newValue);
            insertAfter(value, newValue);
            break;
        case 7:
            deleteFromBeginning();
            break;
        case 8:
            deleteFromEnd();
            break;
        case 9:
            scanf("%d", &value);
            deleteBefore(value);
            break;
        case 10:
            scanf("%d", &value);
            deleteAfter(value);
            break;
        case 11:
```

```
            return 0;
        default:
            printf("Invalid option! Please try again\n");
    }
  }

  return 0;
}
```

Status : Partially correct ***Marks : 0.4/1***

3. Problem Statement

John is working on evaluating polynomials for his math project. He needs to compute the value of a polynomial at a specific point using a singly linked list representation.

Help John by writing a program that takes a polynomial and a value of x as input, and then outputs the computed value of the polynomial.

Example

Input:

2

13

12

11

1

Output:

36

Explanation:

The degree of the polynomial is 2.

Calculate the value of x2: 13 * 12 = 13.

Calculate the value of x1: 12 * 11 = 12.

Calculate the value of x0: 11 * 10 = 11.

Add the values of x2, x1 and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of the degree of the polynomial.

The second line consists of the coefficient x2.

The third line consists of the coefficient of x1.

The fourth line consists of the coefficient x0.

The fifth line consists of the value of x, at which the polynomial should be evaluated.

Output Format

The output is the integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
13
12
11
1
Output: 36

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct Node {
```

```
    int coeff;
    int expo;
    struct Node* next;
} Node;

// Create a new term (node)
Node* createNode(int coeff, int expo) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->coeff = coeff;
    newNode->expo = expo;
    newNode->next = NULL;
    return newNode;
}

// Insert node at end
Node* insertEnd(Node* head, int coeff, int expo) {
    Node* newNode = createNode(coeff, expo);
    if (head == NULL)
        return newNode;

    Node* temp = head;
    while (temp->next)
        temp = temp->next;

    temp->next = newNode;
    return head;
}

// Evaluate the polynomial at x
int evaluatePolynomial(Node* head, int x) {
    int result = 0;
    Node* temp = head;
    while (temp) {
        result += temp->coeff * pow(x, temp->expo);
        temp = temp->next;
    }
    return result;
}

// Main function
int main() {
    int degree, x, coeff;
```

```
    scanf("%d", &degree);

    Node* poly = NULL;

    // Read coefficients starting from highest degree
    for (int i = degree; i >= 0; i--) {
        scanf("%d", &coeff);
        poly = insertEnd(poly, coeff, i);
    }

    scanf("%d", &x); // value of x

    int result = evaluatePolynomial(poly, x);
    printf("%d\n", result);

    return 0;
}
```

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

4. Problem Statement

Imagine you are managing the backend of an e-commerce platform. Customers place orders at different times, and the orders are stored in two separate linked lists. The first list holds the orders from morning, and the second list holds the orders from the evening.

Your task is to merge the two lists so that the final list holds all orders in sequence from the morning list followed by the evening orders, in the same order

Input Format

The first line contains an integer n , representing the number of orders in the morning list.

The second line contains n space-separated integers representing the morning orders.

The third line contains an integer m , representing the number of orders in the evening list.

The fourth line contains m space-separated integers representing the evening orders.

Output Format

The output should be a single line containing space-separated integers representing the merged order list, with morning orders followed by evening orders.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
101 102 103
2
104 105
Output: 101 102 103 104 105

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int order_id;
    struct Node* next;
} Node;

// Create new node
Node* createNode(int order_id) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->order_id = order_id;
    newNode->next = NULL;
    return newNode;
}

// Append node to the end
Node* append(Node* head, Node** tail, int order_id) {
    Node* newNode = createNode(order_id);
```

```
    if (!head) {
        *tail = newNode;
        return newNode;
    }
    (*tail)->next = newNode;
    *tail = newNode;
    return head;
}

// Print the merged list
void printList(Node* head) {
    Node* temp = head;
    while (temp) {
        printf("%d ", temp->order_id);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    int n, m, order_id;
    Node *morningHead = NULL, *morningTail = NULL;
    Node *eveningHead = NULL, *eveningTail = NULL;

    // Read morning orders
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &order_id);
        morningHead = append(morningHead, &morningTail, order_id);
    }

    // Read evening orders
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d", &order_id);
        eveningHead = append(eveningHead, &eveningTail, order_id);
    }

    // Merge evening list after morning list
    if (morningTail)
        morningTail->next = eveningHead;
```

```
    else
        morningHead = eveningHead; // morning list is empty

    // Print merged list
    printList(morningHead);

    return 0;
}
```

Status : Correct *Marks : 1/1*

5. Problem Statement

Bharath is very good at numbers. As he is piled up with many works, he decides to develop programs for a few concepts to simplify his work. As a first step, he tries to arrange even and odd numbers using a linked list. He stores his values in a singly-linked list.

Now he has to write a program such that all the even numbers appear before the odd numbers. Finally, the list is printed in such a way that all even numbers come before odd numbers. Additionally, the even numbers should be in reverse order, while the odd numbers should maintain their original order.

Example

Input:

6

3 1 0 4 30 12

Output:

12 30 4 0 3 1

Explanation:

Even elements: 0 4 30 12

Reversed Even elements: 12 30 4 0

Odd elements: 3 1

So the final list becomes: 12 30 4 0 3 1

Input Format

The first line consists of an integer n representing the size of the linked list.

The second line consists of n integers representing the elements separated by space.

Output Format

The output prints the rearranged list separated by a space.

The list is printed in such a way that all even numbers come before odd numbers and the even numbers should be in reverse order, while the odd numbers should maintain their original order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
3 1 0 4 30 12

Output: 12 30 4 0 3 1

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

// Create new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
```

```
    newNode->next = NULL;
    return newNode;
}

// Push at head (used for even list to reverse while inserting)
Node* push(Node* head, int data) {
    Node* newNode = createNode(data);
    newNode->next = head;
    return newNode;
}

// Append at end (used for odd list to maintain order)
Node* append(Node* tail, int data) {
    Node* newNode = createNode(data);
    if (tail != NULL)
        tail->next = newNode;
    return newNode;
}

// Print list
void printList(Node* head) {
    Node* temp = head;
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

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

    Node *evenList = NULL, *oddList = NULL, *oddTail = NULL;

    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        if (val % 2 == 0) {
            evenList = push(evenList, val); // reverse insertion
        } else {
            if (!oddList) {
                oddList = append(NULL, val);
```

```
            oddTail = oddList;
        } else {
            oddTail = append(oddTail, val);
        }
    }
}

// Merge even and odd lists
if (!evenList) {
    printList(oddList);
} else {
    Node* temp = evenList;
    while (temp->next)
        temp = temp->next;
    temp->next = oddList;
    printList(evenList);
}

return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Pranav wants to clockwise rotate a doubly linked list by a specified number of positions. He needs your help to implement a program to achieve this. Given a doubly linked list and an integer representing the number of positions to rotate, write a program to rotate the list clockwise.

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 linked list elements.

The third line consists of an integer k, representing the number of places to rotate the list.

Output Format

The output displays the elements of the doubly linked list after rotating it by k positions.

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


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


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


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

    Node* temp = head;
```

```
    while (temp->next)
        temp = temp->next;

    temp->next = newNode;
    newNode->prev = temp;
    return head;
}


Node* rotateClockwise(Node* head, int k, int n) {
    if (k == 0 || k >= n) return head;


    Node* last = head;
    while (last->next)
        last = last->next;


    last->next = head;
    head->prev = last;


    int steps = n - k;
    Node* newTail = head;
    for (int i = 0; i < steps - 1; i++) {
        newTail = newTail->next;
    }

    Node* newHead = newTail->next;


    newHead->prev = NULL;
    newTail->next = NULL;

    return newHead;
}


void printList(Node* head) {
    Node* temp = head;
    while (temp) {
        printf("%d ", temp->data);
```

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


int main() {
    int n, k, val;
    Node* head = NULL;


    scanf("%d", &n);


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

    // Read k
    scanf("%d", &k);

    // Rotate the list
    head = rotateClockwise(head, k, n);

    // Print result
    printList(head);

    return 0;
}
```

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

2. Problem Statement

Bala is a student learning about the doubly linked list and its functionalities. He came across a problem where he wanted to create a doubly linked list by appending elements to the front of the list.

After populating the list, he wanted to delete the node at the given position

from the beginning. Write a suitable code to help Bala.

Input Format

The first line contains an integer N, the number of elements in the doubly linked list.

The second line contains N integers separated by a space, the data values of the nodes in the doubly linked list.

The third line contains an integer X, the position of the node to be deleted from the doubly linked list.

Output Format

The first line of output displays the original elements of the doubly linked list, separated by a space.

The second line prints the updated list after deleting the node at the given position X from the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
2
Output: 50 40 30 20 10
50 30 20 10

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

Node* appendFront(Node* head, int data) {
    Node* newNode = createNode(data);
    if (head == NULL) {
        return newNode;
    }
    newNode->next = head;
    head->prev = newNode;
    return newNode;
}

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

Node* deleteAtPosition(Node* head, int position) {
    if (head == NULL) {
        return NULL;
    }

    Node* temp = head;

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

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

    if (temp == NULL) {
        return head;
    }

    if (temp->next == NULL) {
        temp->prev->next = NULL;
    } else {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
    }

    free(temp);
    return head;
}

int main() {
    int N, X;
    int data;

    scanf("%d", &N);

    Node* head = NULL;

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

    scanf("%d", &X);

    printList(head);

    head = deleteAtPosition(head, X);

    printList(head);

    return 0;
```

}

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

3. Problem Statement

Rohan is a software developer who is working on an application that processes data stored in a Doubly Linked List. He needs to implement a feature that finds and prints the middle element(s) of the list. If the list contains an odd number of elements, the middle element should be printed. If the list contains an even number of elements, the two middle elements should be printed.

Help Rohan by writing a program that reads a list of numbers, prints the list, and then prints the middle element(s) based on the number of elements in the list.

Input Format

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

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

Output Format

The first line prints the elements of the list separated by space. (There is an extra space at the end of this line.)

The second line prints the middle element(s) based on the number of elements.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
20 52 40 16 18
Output: 20 52 40 16 18

40

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

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

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

```
void printMiddle(Node* head, int n) {
    if (head == NULL) {
        return;
    }

    Node* slow = head;
    Node* fast = head;
    int count = 0;

    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
        count += 2;
    }


    if (n % 2 == 1) {
        printf("%d\n", slow->data);
    } else {

        printf("%d %d\n", slow->prev->data, slow->data);
    }
}

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

    Node* head = NULL;
    int data;


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


    printList(head);
```

```
    printMiddle(head, n);

    return 0;
}
```

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

4. Problem Statement

Riya is developing a contact management system where recently added contacts should appear first. She decides to use a doubly linked list to store contact IDs in the order they are added. Initially, new contacts are inserted at the front of the list. However, sometimes she needs to insert a new contact at a specific position in the list based on priority.

Help Riya implement this system by performing the following operations:

Insert contact IDs at the front of the list as they are added.Insert a new contact at a given position in the list.

Input Format

The first line of input consists of an integer N, representing the initial size of the linked list.

The second line consists of N space-separated integers, representing the values of the linked list to be inserted at the front.

The third line consists of an integer position, representing the position at which the new value should be inserted (position starts from 1).

The fourth line consists of integer data, representing the new value to be inserted.

Output Format

The first line of output prints the original list after inserting initial elements to the front.

The second line prints the updated linked list after inserting the element at the specified position.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 20 30 40
3
25

Output: 40 30 20 10
40 30 25 20 10

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

Node* insertAtFront(Node* head, int data) {
    Node* newNode = createNode(data);
    if (head == NULL)
        return newNode;
    newNode->next = head;
    head->prev = newNode;
    return newNode;
}

void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
```

```
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

Node* insertAtPosition(Node* head, int position, int data) {
    Node* newNode = createNode(data);
    if (position == 1) {
        newNode->next = head;
        if (head != NULL)
            head->prev = newNode;
        return newNode;
    }

    Node* temp = head;
    for (int i = 1; i < position - 1 && temp != NULL; i++)
        temp = temp->next;

    if (temp == NULL || temp->next == NULL) {

        temp->next = newNode;
        newNode->prev = temp;
    } else {
        newNode->next = temp->next;
        newNode->prev = temp;
        temp->next->prev = newNode;
        temp->next = newNode;
    }

    return head;
}

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

    Node* head = NULL;
    int value;

    for (int i = 0; i < N; i++) {
        scanf("%d", &value);
```

```
        head = insertAtFront(head, value);
    }

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

    printList(head);

    head = insertAtPosition(head, position, data);

    printList(head);

    return 0;
}
```

Status : Correct *Marks : 10/10*

5. Problem Statement

Tom is a software developer working on a project where he has to check if a doubly linked list is a palindrome. He needs to write a program to solve this problem. Write a program to help Tom check if a given doubly linked list is a palindrome or not.

Input Format

The first line 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 linked list elements.

Output Format

The first line displays the space-separated integers, representing the doubly linked list.

The second line displays one of the following:

1. If the doubly linked list is a palindrome, print "The doubly linked list is a palindrome".
2. If the doubly linked list is not a palindrome, print "The doubly linked list is not a palindrome".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 2 1

Output: 1 2 3 2 1
The doubly linked list is a palindrome

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

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

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

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

    temp->next = newNode;
    newNode->prev = temp;
    return head;
```

```
}

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

int isPalindrome(Node* head) {
    if (head == NULL)
        return 1;

    Node* start = head;
    Node* end = head;

    while (end->next != NULL)
        end = end->next;

    while (start != end && start->prev != end) {
        if (start->data != end->data)
            return 0;
        start = start->next;
        end = end->prev;
    }

    return 1;
}

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

    Node* head = NULL;

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

```
    printList(head);

    if (isPalindrome(head))
        printf("The doubly linked list is a palindrome\n");
    else
        printf("The doubly linked list is not a palindrome\n");

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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) {
   top++;
   items[top]=value;
   printf("Pushed: %c\n",value);
   return;
}

void pop() {
   if(isEmpty()){
      printf("Stack is empty. Nothing to pop.\n");
   }
   else{
      printf("Popped: %c\n",items[top]);
      top--;
   }
}
void display() {
   if(isEmpty()){
      printf("Stack is empty.\n");
      return;
   }

   printf("Stack elements: ");
   for(int i=top;i>=0;i--){
      printf("%c ",items[i]);
```

```
    }
    printf("\n");
}
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: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 : 10

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>
int main(){
    int arr[100];
    int top=-1;
   while(1){
      int n;
      scanf("%d",&n);
      switch (n){
         case 1:
         {
          int val;
          scanf(" %d",&val);
          top++;
          arr[top]=val;
          printf("Book ID %d is pushed onto the stack\n",val);
          break;
         }

          case 2:
             if(top==-1){
                printf("Stack Underflow\n");
                break;
             }
             printf("Book ID %d is popped from the stack\n",arr[top]);
             top=top-1;
             break;
          case 3:
             if(top==-1){
                printf("Stack is empty\n");
                break;
             }
             printf("Book ID in the stack: ");
             for(int i=top;i>=0;i--){
                printf("%d ",arr[i]);
             }
```

```
            printf("\n");
            break;

        case 4:
            printf("Exiting the program\n");
            return 0;

        default:
            printf("Invalid choice\n");
    }
  }
return  0;
}//main
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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));
    newnode->data=value;
    newnode->next=top;
    top=newnode;
    printf("Pushed element: %d\n",value);
    return;
}
```

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

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

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: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

Answer

Pop

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

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

Answer

All of the mentioned options

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

3. 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***

4. 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***

5. 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***

6. 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***

7. 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***

8. 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***

9. 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***

10. 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***

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

Answer

142

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
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***

14. 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***

15. 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***

16. 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***

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

Answer

-18

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

18. 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

Stack operations will be performed smoothly

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

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

Answer

Efficient memory usage

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

20. 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

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 17

Section 1 : MCQ

1. Which of the following can be used to delete an element from the front end of the queue?

Answer

None of these

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

2. Which of the following properties is associated with a queue?

Answer

First In First Out

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

3. What are the applications of dequeue?

Answer

All the mentioned options

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

4. Which one of the following is an application of Queue Data Structure?

Answer

All of the mentioned options

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

5. Which operations are performed when deleting an element from an array-based queue?

Answer

Dequeue

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

6. In a linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a non-empty queue?

Answer

Only rear pointer

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

7. The essential condition that is checked before insertion in a queue is?

Answer

Overflow

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

8. The process of accessing data stored in a serial access memory is similar to manipulating data on a

Answer

Stack

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

9. What is the functionality of the following piece of code?

```
public void function(Object item)
{
   Node temp=new Node(item,trail);
   if(isEmpty())
   {
      head.setNext(temp);
      temp.setNext(trail);
   }
   else
   {
      Node cur=head.getNext();
      while(cur.getNext()!=trail)
      {
         cur=cur.getNext();
      }
      cur.setNext(temp);
   }
   size++;
}
```

Answer

Insert at the rear end of the dequeue

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

10. What does the front pointer in a linked list implementation of a queue contain?

Answer

The address of the first element

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

11. When new data has to be inserted into a stack or queue, but there is no available space. This is known as

Answer

overflow

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

12. A normal queue, if implemented using an array of size MAX_SIZE, gets full when

Answer

Rear = MAX_SIZE – 1

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

13. What will the output of the following code?

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int* arr;
    int front;
    int rear;
    int size;
} Queue;
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->arr = (int*)malloc(5 * sizeof(int));
    queue->front = 0;
    queue->rear = -1;
    queue->size = 0;
```

```
    return queue;
}
int main() {
    Queue* queue = createQueue();
    printf("%d", queue->size);
    return 0;
}
```

Answer

0

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

14. After performing this set of operations, what does the final list look to contain?

```
InsertFront(10);
InsertFront(20);
InsertRear(30);
DeleteFront();
InsertRear(40);
InsertRear(10);
DeleteRear();
InsertRear(15);
display();
```

Answer

20 30 40 15

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

15. Insertion and deletion operation in the queue is known as

Answer

Enqueue and Dequeue

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

16. In what order will they be removed If the elements “A”, “B”, “C” and “D” are placed in a queue and are deleted one at a time

Answer

ABCD

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

17. Front and rear pointers are tracked in the linked list implementation of a queue. Which of these pointers will change during an insertion into the EMPTY queue?

Answer

Both front and rear pointer

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

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

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 5
typedef struct {
    int* arr;
    int front;
    int rear;
    int size;
} Queue;
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->arr = (int*)malloc(MAX_SIZE * sizeof(int));
    queue->front = -1;
    queue->rear = -1;
    queue->size = 0;
    return queue;
}
int isEmpty(Queue* queue) {
    return (queue->size == 0);
```

```
}
int main() {
    Queue* queue = createQueue();
    printf("Is the queue empty? %d", isEmpty(queue));
    return 0;
}
```

Answer

Is the queue empty? 1

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

19. In linked list implementation of a queue, the important condition for a queue to be empty is?

Answer

FRONT is null

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

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

```
#include <stdio.h>
#define MAX_SIZE 5
typedef struct {
    int arr[MAX_SIZE];
    int front;
    int rear;
    int size;
} Queue;

void enqueue(Queue* queue, int data) {
    if (queue->size == MAX_SIZE) {
        return;
    }
    queue->rear = (queue->rear + 1) % MAX_SIZE;
    queue->arr[queue->rear] = data;
    queue->size++;
```

```
}
int dequeue(Queue* queue) {
    if (queue->size == 0) {
        return -1;
    }
    int data = queue->arr[queue->front];
    queue->front = (queue->front + 1) % MAX_SIZE;
    queue->size--;
    return data;
}
int main() {
    Queue queue;
    queue.front = 0;
    queue.rear = -1;
    queue.size = 0;
    enqueue(&queue, 1);
    enqueue(&queue, 2);
    enqueue(&queue, 3);
    printf("%d ", dequeue(&queue));
    printf("%d ", dequeue(&queue));
    enqueue(&queue, 4);
    enqueue(&queue, 5);
    printf("%d ", dequeue(&queue));
    printf("%d ", dequeue(&queue));
    return 0;
}
```

Answer

1 2 3 4

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are tasked with implementing basic operations on a queue data structure using a linked list.

You need to write a program that performs the following operations on a queue:

Enqueue Operation: Implement a function that inserts an integer element at the rear end of the queue.Print Front and Rear: Implement a function that prints the front and rear elements of the queue. Dequeue Operation: Implement a function that removes the front element from the queue.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the queue.

The second line consists of N space-separated integers, representing the queue elements.

Output Format

The first line prints "Front: X, Rear: Y" where X is the front and Y is the rear elements of the queue.

The second line prints the message indicating that the dequeue operation (front element removed) is performed: "Performing Dequeue Operation:".

The last line prints "Front: M, Rear: N" where M is the front and N is the rear elements after the dequeue operation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
12 56 87 23 45

Output: Front: 12, Rear: 45
Performing Dequeue Operation:
Front: 56, Rear: 45

Answer

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

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

struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int d) {
   struct Node *newnode=(struct Node*)malloc(sizeof(struct Node));
   newnode->data=d;
   newnode->next=NULL;
```

```
    if(front==NULL){
        front=newnode;
        rear=newnode;
        return;
    }
    rear->next=newnode;
    rear=newnode;
    return;

}

void printFrontRear() {
    printf("Front: %d, Rear: %d\n",front->data,rear->data);
    return;
}

void dequeue() {
    struct Node *temp=front;
    front=front->next;
    free(temp);
    return;
}

int main() {
    int n, data;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(data);
    }
    printFrontRear();
    printf("Performing Dequeue Operation:\n");
    dequeue();
    printFrontRear();
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;
}

// You are using GCC
#include<ctype.h>
int isOperand(char ch) {
    return isalpha(ch);
}

int Prec(char ch) {
    switch (ch){
        case '+':
            return 1;
        case '-':
            return 1;

        case '*':
            return 2;
```

```
        case '/':
            return 2;
        case '^':
            return 3;
    }
    return 0;
}

void infixToPostfix(char* exp) {
    struct Stack *stack=createStack(100);
    for(int i=0; exp[i]!='\0' ;i++){
        char ch=exp[i];

        if (isOperand(ch)){
            printf("%c",ch);
        }

        else if(ch=='('){
            push(stack,ch);
        }
        else if(ch==')'){
            while(!isEmpty(stack) &&  peek(stack)!='('){
                printf("%c",pop(stack));
            }

            pop(stack);
        }
        else{
            while(!isEmpty(stack)&& Prec(peek(stack))>=Prec(ch)){
                char val2 =pop(stack);
                printf("%c",val2);
            }
            push(stack,ch);
        }

    }//for
        while(!isEmpty(stack)){
            printf("%c",pop(stack));
        }
}//main
int main() {
    char exp[100];
```

```
    scanf("%s", exp);

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

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In an office setting, a print job management system is used to efficiently handle and process print jobs. The system is implemented using a queue data structure with an array.

The program provides the following operations:

Enqueue Print Job: Add a print job with a specified number of pages to the end of the queue.Dequeue Print Job: Remove and process the next print job in the queue.Display Queue: Display the print jobs in the queue

The program should ensure that print jobs are processed in the order they are received.

Input Format

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

Choice 1: Enqueue the print job into the queue. If the choice is 1, the following input is a space-separated integer, representing the pages to be enqueued into the queue.

Choice 2: Dequeue a print job from the queue.

Choice 3: Display the print jobs in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

1. Insert the given page into the queue and display "Print job with [page] pages is enqueued." where [page] is the number of pages that are inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a page from the queue and display "Processing print job: [page] pages" where [page] is the corresponding page that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Print jobs in the queue: " followed by the space-separated pages present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
10
1
20
1
30
1
40
1
50
1
60
3
2
3
4

Output: Print job with 10 pages is enqueued.
Print job with 20 pages is enqueued.
Print job with 30 pages is enqueued.
Print job with 40 pages is enqueued.
Print job with 50 pages is enqueued.
Queue is full. Cannot enqueue.
Print jobs in the queue: 10 20 30 40 50
Processing print job: 10 pages
Print jobs in the queue: 20 30 40 50
Exiting program

Answer

```
void enqueue(int pages) {
    if (rear == MAX_SIZE - 1) {
        printf("Queue is full. Cannot enqueue.\n");
        return;
    }
```

```
    if (front == -1) {
        front = 0;
    }

    rear++;
    queue[rear] = pages;
    printf("Print job with %d pages is enqueued.\n", pages);
}


void dequeue() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return;
    }

    printf("Processing print job: %d pages\n", queue[front]);
    front++;


    if (front > rear) {
        front = rear = -1;
    }
}


void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return;
    }

    printf("Print jobs in the queue: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Write a program to implement a queue using an array and pointers. The program should provide the following functionalities:

Insert an element into the queue.Delete an element from the queue.Display the elements in the queue.

The queue has a maximum capacity of 5 elements. If the queue is full and an insertion is attempted, a "Queue is full" message should be displayed. If the queue is empty and a deletion is attempted, a "Queue is empty" message should be displayed.

Input Format

Each line contains an integer representing the chosen option from 1 to 3.

Option 1: Insert an element into the queue followed by an integer representing the element to be inserted, separated by a space.

Option 2: Delete an element from the queue.

Option 3: Display the elements in the queue.

Output Format

For option 1 (insertion):-

1. The program outputs: "<data> is inserted in the queue." if the data is successfully inserted.
2. "Queue is full." if the queue is already full and cannot accept more elements.

For option 2 (deletion):-

1. The program outputs: "Deleted number is: <data>" if an element is successfully deleted and returns the value of the deleted element.
2. "Queue is empty." if the queue is empty no elements can be deleted.

For option 3 (display):-

1. The program outputs: "Elements in the queue are: <element1> <element2> ... <elementN>" where <element1>, <element2>, ..., <elementN> represent the elements present in the queue.
2. "Queue is empty." if the queue is empty no elements can be displayed.

For invalid options, the program outputs: "Invalid option."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 10

3
5
Output: 10 is inserted in the queue.
Elements in the queue are: 10
Invalid option.

Answer

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

#define max 5

int queue[max];
int front = -1, rear = -1;

int insertq(int *data)
{
   if (rear==max-1){
      return 0;
   }

   if(front==-1){
      front=0;
   }
   rear++;
   queue[rear]=*data;
   return 1;
}

int delq()
{
  if(front==-1 || front>rear){
     printf("Queue is empty.\n");
   return 0;
  }

  printf("Deleted number is: %d\n",queue[front]);
  front++;
  return 0;

}
```

```
void display()
{
   if(front==-1 || front>rear){
      printf("Queue is empty.\n");
      return;
   }
   printf("Elements in the queue are:");
   for(int i =front;i<=rear;i++){
      printf("%d ",queue[i]);
   }
   printf("\n");
}
int main()
{
   int data, reply, option;
   while (1)
   {
      if (scanf("%d", &option) != 1)
         break;
      switch (option)
      {
         case 1:
            if (scanf("%d", &data) != 1)
               break;
            reply = insertq(&data);
            if (reply == 0)
               printf("Queue is full.\n");
            else
               printf("%d is inserted in the queue.\n", data);
            break;
         case 2:
            delq();  //    Called without arguments
            break;
         case 3:
            display();
            break;
         default:
            printf("Invalid option.\n");
            break;
      }
   }
   return 0;
```

}

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a bustling IT department, staff regularly submit helpdesk tickets to request technical assistance. Managing these tickets efficiently is vital for providing quality support.

Your task is to develop a program that uses an array-based queue to handle and prioritize helpdesk tickets based on their unique IDs.

Implement a program that provides the following functionalities:

Enqueue Helpdesk Ticket: Add a new helpdesk ticket to the end of the queue. Provide a positive integer representing the ticket ID for the new ticket.Dequeue Helpdesk Ticket: Remove and process the next helpdesk ticket from the front of the queue. The program will display the ticket ID of the processed ticket.Display Queue: Display the ticket IDs of all the

helpdesk tickets currently in the queue.

Input Format

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

Choice 1: Enqueue the ticket ID into the queue. If the choice is 1, the following input is a space-separated integer, representing the ticket ID to be enqueued into the queue.

Choice 2: Dequeue a ticket from the queue.

Choice 3: Display the ticket IDs in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

1. Insert the given ticket ID into the queue and display "Helpdesk Ticket ID [id] is enqueued." where [id] is the ticket ID that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a ticket ID from the queue and display "Dequeued Helpdesk Ticket ID: " followed by the corresponding ID that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Helpdesk Ticket IDs in the queue are: " followed by the space-separated ticket IDs present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting the program"

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 101
1 202
1 203
1 204
1 205
1 206
3
2
3
4

Output: Helpdesk Ticket ID 101 is enqueued.
Helpdesk Ticket ID 202 is enqueued.
Helpdesk Ticket ID 203 is enqueued.
Helpdesk Ticket ID 204 is enqueued.
Helpdesk Ticket ID 205 is enqueued.
Queue is full. Cannot enqueue.
Helpdesk Ticket IDs in the queue are: 101 202 203 204 205
Dequeued Helpdesk Ticket ID: 101
Helpdesk Ticket IDs in the queue are: 202 203 204 205
Exiting the program

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

int ticketIDs[MAX_SIZE];
int front = -1;
int rear = -1;
int lastDequeued;

void initializeQueue() {
    front = -1;
    rear = -1;
}
```

```
// You are using GCC
int isEmpty() {

    return front>rear ||front==-1;
}

int isFull() {
    return rear==MAX_SIZE-1;
}

int enqueue(int ticketID) {
     if( isFull()){
         printf("Queue is full. cannot enqueue.\n");
         return 0;
     }
    if(front==-1){
        front=0;
    }
    rear++;
    ticketIDs[rear]=ticketID;
    printf("Helpdesk Ticket ID %d is enqueued.\n",ticketID);
    return 0;

}

int dequeue() {
    if (isEmpty()){
        return 0;
    }
    lastDequeued=ticketIDs[front];
    front++;
    return 1;

}

void display() {
    if(isEmpty()){
        printf("Queue is empty.\n");
        return;
    }
    printf("Helpdesk Ticket IDs in the queue are:");
    int temp=front;
```

```
    for (int i = front; i<=rear;i++){
        printf("%d ",ticketIDs[i]);
    }
    printf("\n");

}
int main() {
    int ticketID;
    int option;
    initializeQueue();
    while (1) {
        if (scanf("%d", &option) == EOF) {
            break;
        }
        switch (option) {
            case 1:
                if (scanf("%d", &ticketID) == EOF) {
                    break;
                }
                enqueue(ticketID);
                break;
            case 2:
                if (dequeue()) {
                    printf("Dequeued Helpdesk Ticket ID: %d\n", lastDequeued);
                } else {
                    printf("Queue is empty.\n");
                }
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting the program\n");
                return 0;
            default:
                printf("Invalid option.\n");
                break;
        }
    }
    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine a bustling coffee shop, where customers are placing their orders for their favorite coffee drinks. The cafe owner Sheeren wants to efficiently manage the queue of coffee orders using a digital system. She needs a program to handle this queue of orders.

You are tasked with creating a program that implements a queue for coffee orders. Each character in the queue represents a customer's coffee order, with 'L' indicating a latte, 'E' indicating an espresso, 'M' indicating a macchiato, 'O' indicating an iced coffee, and 'N' indicating a nabob.

Customers can place orders and enjoy their delicious coffee drinks.

Input Format

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

Choice 1: Enqueue the coffee order into the queue. If the choice is 1, the following input is a space-separated character ('L', 'E', 'M', 'O', 'N').

Choice 2: Dequeue a coffee order from the queue.

Choice 3: Display the orders in the queue.

Choice 4: Exit the program.

Output Format

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

If the choice is 1:

1. Insert the given order into the queue and display "Order for [order] is enqueued." where [order] is the coffee order that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue more orders."

If the choice is 2:

1. Dequeue a character from the queue and display "Dequeued Order: " followed by the corresponding order that is dequeued.
2. If the queue is empty without any orders, print "No orders in the queue."

If the choice is 3:

1. The output prints "Orders in the queue are: " followed by the space-separated orders present in the queue.
2. If there are no orders in the queue, print "Queue is empty. No orders available."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

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

Sample Test Case

Input: 1 L
1 E
1 M
1 O
1 N
1 O
3
2
3
4

Output: Order for L is enqueued.
Order for E is enqueued.
Order for M is enqueued.
Order for O is enqueued.
Order for N is enqueued.
Queue is full. Cannot enqueue more orders.
Orders in the queue are: L E M O N
Dequeued Order: L
Orders in the queue are: E M O N
Exiting program

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

char orders[MAX_SIZE];
int front = -1;
int rear = -1;

void initializeQueue() {
    front = -1;
    rear = -1;
}
// You are using GCC
```

```
int isEmpty() {
    return front==-1;
    //Type your code here
}

int isFull() {
    return(rear+1)%MAX_SIZE==front;
    //Type your code here
}

int enqueue(char order) {
    if(isFull()){
        printf("Queue is full. Cannot enqueue more orders.\n");
        return 0;
    }
    if(isEmpty()){
        front=rear=0;
    }
    else{
        rear=(rear+1)%MAX_SIZE;
    }
    orders[rear]=order;
    printf("Order for %c is enqueued.\n",order);
    return 1;
    //Type your code here
}

int dequeue() {
    if(isEmpty()){
        printf("No orders in the queue.\n");
        return 0;
    }
    char order=orders[front];
    if(front==rear){
        front=rear=-1;
    }
    else{
        front=(front+1)%MAX_SIZE;

    }
    printf("Dequeued Order: %c\n",order);
    return 1;
```

```
    //Type your code here
}

void display() {
    if(isEmpty()){
        printf("Queue is empty. No orders available.\n");
        return;
    }
    printf("Orders in the queue are: ");
    int i=front;
    while(1){
        printf("%c",orders[i]);
        if(i==rear)
            break;
        i=(i+1)%MAX_SIZE;
    }
    printf("\n");
    //Type your code here
}

int main() {
    char order;
    int option;
    initializeQueue();
    while (1) {
        if (scanf("%d", &option) != 1) {
            break;
        }
        switch (option) {
            case 1:
                if (scanf(" %c", &order) != 1) {
                    break;
                }
                if (enqueue(order)) {
                }
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
```

```
            printf("Exiting program");
            return 0;
        default:
            printf("Invalid option.\n");
            break;
    }
  }
  return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are tasked with implementing basic operations on a queue data structure using a linked list.

You need to write a program that performs the following operations on a queue:

Enqueue Operation: Implement a function that inserts an integer element at the rear end of the queue.Print Front and Rear: Implement a function that prints the front and rear elements of the queue. Dequeue Operation: Implement a function that removes the front element from the queue.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the queue.

The second line consists of N space-separated integers, representing the queue elements.

Output Format

The first line prints "Front: X, Rear: Y" where X is the front and Y is the rear elements of the queue.

The second line prints the message indicating that the dequeue operation (front element removed) is performed: "Performing Dequeue Operation:".

The last line prints "Front: M, Rear: N" where M is the front and N is the rear elements after the dequeue operation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
12 56 87 23 45

Output: Front: 12, Rear: 45
Performing Dequeue Operation:
Front: 56, Rear: 45

Answer

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

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

struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int d) {
   struct Node *newnode=(struct Node*)malloc(sizeof(struct Node));
   newnode->data=d;
   newnode->next=NULL;
```

```
    if(front==NULL){
        front=newnode;
        rear=newnode;
        return;
    }
    rear->next=newnode;
    rear=newnode;
    return;

}

void printFrontRear() {
    printf("Front: %d, Rear: %d\n",front->data,rear->data);
    return;
}

void dequeue() {
    struct Node *temp=front;
    front=front->next;
    free(temp);
    return;
}

int main() {
    int n, data;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(data);
    }
    printFrontRear();
    printf("Performing Dequeue Operation:\n");
    dequeue();
    printFrontRear();
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 pre-order traversal of the given binary search tree.

Answer

13, 2, 1, 4, 14, 18

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

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

Answer

1, 4, 2, 18, 14, 13

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

3. 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***

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

Answer

14

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

5. 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***

6. 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***

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

Answer

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

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

8. 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***

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

Answer

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

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

10. 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***

11. 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***

12. 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***

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

Answer

1, 2, 4, 13, 14, 18

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

14. 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***

15. 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***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 : 10

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;
}

struct TreeNode* insert(struct TreeNode* root, int value) {
    if (root == NULL) {
        root = (struct TreeNode*)malloc(sizeof(struct TreeNode));
```

```
        root->data = value;
        root->left = root->right = NULL;
        return root;
    }

    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);
    return root;
}

struct TreeNode* findMin(struct TreeNode* root) {
    while (root->left != NULL)
        root = root->left;
    return root;
}

struct TreeNode* deleteNode(struct TreeNode* root, int value) {
    if (root == NULL)
        return NULL;

    if (value < root->data)
        root->left = deleteNode(root->left, value);
    else if (value > root->data)
        root->right = deleteNode(root->right, value);
    else {
        if (root->left == NULL) {
            struct TreeNode* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct TreeNode* temp = root->left;
            free(root);
            return temp;
        }
        struct TreeNode* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }

    return root;
```

```
}
void inorderTraversal(struct TreeNode* root) {
    if (root == NULL)
        return;
    inorderTraversal(root->left);
    printf("%d ", root->data);
    inorderTraversal(root->right);
}
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 : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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) {
    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;
    //Type your code here
```

```
}
void printPreorder(struct Node* root) {
    if(root!=NULL)
    {
        printf("%d\t",root->data);
        printPreorder(root->left);
        printPreorder(root->right);
    }
    //Type your code here
}
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: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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 : 10

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
#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;
}

struct Node* insert(struct Node* root, int value) {
    if (root == NULL) {
```

```
        return createNode(value);
    }
    if (value < root->data) {
        root->left = insert(root->left, value);
    } else if (value > root->data) {
        root->right = insert(root->right, value);
    }
    return root;
}

int search(struct Node* root, int key) {
    if (root == NULL) {
        return 0;
    }
    if (key == root->data) {
        return 1;
    } else if (key < root->data) {
        return search(root->left, key);
    } else {
        return search(root->right, key);
    }
}

int main() {
    int n, key, value;
    scanf("%d", &n);
    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insert(root, value);
    }
    scanf("%d", &key);
    if (search(root, key)) {
        printf("Value %d is found in the tree.\n", key);
    } else {
        printf("Value %d is not found in the tree.\n", key);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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) {
    if(root==NULL)
    {
        struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
        newNode->data=data;
        newNode->left=NULL;
        newNode->right=NULL;
```

```
        root=newNode;
    }
    else if(data < root->data)
        root->left=insert(root->left,data);
    else if(data > root->data)
        root->right=insert(root->right,data);
    return root;
    //Type your code here
}

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

int findMinValue(struct Node* root) {
    if(root == NULL)
        printf("room is empty");
    else if(root->left==NULL)
        return root->data;
    else
        return findMinValue(root->left);
    //Type your code here
}
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: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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) {
    if(root ==NULL){
        struct TreeNode* newNode=(struct TreeNode*)malloc(sizeof(struct
TreeNode));
        newNode->data=key;
        newNode->left=NULL;
        newNode->right=NULL;
        root=newNode;
    }
    else if(key < root->data)
        root->left=insert(root->left,key);
    else if(key > root->data)
```

```
        root->right=insert(root->right,key);
    return root;
    //Type your code here
}

int findMax(struct TreeNode* root) {
    if(root==NULL)
      printf("Tree is empty");
    else if(root->right==NULL)
        return root->data;
    else
        return findMax(root->right);


    //Type your code here
}
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***

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_MCQ_Updated_1

Attempt : 1
Total Mark : 20
Marks Obtained : 18

Section 1 : MCQ

1. Which of the following methods is used for sorting in merge sort?

Answer

merging

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

2. Why is Merge Sort preferred for sorting large datasets compared to Quick Sort?

Answer

Merge Sort has better worst-case time complexity

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

3. Which of the following sorting algorithms is based on the divide and conquer method?

Answer

Merge Sort

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

4. Let P be a quick sort program to sort numbers in ascending order using the first element as a pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2}, respectively. Which one of the following holds?

Answer

t1 > t2

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

5. Merge sort is _____.

Answer

Comparison-based sorting algorithm

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

6. Which of the following scenarios is Merge Sort preferred over Quick Sort?

Answer

When sorting linked lists

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

7. The following code snippet is an example of a quick sort. What do the 'low' and 'high' parameters represent in this code?

```
void quickSort(int arr[], int low, int high) {
```

```
    if (low < high) {
        int pivot = partition(arr, low, high);
        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}
```

Answer

The range of elements to sort within the array

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

8. What happens when Merge Sort is applied to a single-element array?

Answer

The array remains unchanged and no merging is required

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

9. What is the best sorting algorithm to use for the elements in an array that are more than 1 million in general?

Answer

Quick sort.

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

10. Which of the following strategies is used to improve the efficiency of Quicksort in practical implementations?

Answer

Choosing the pivot randomly or using the median-of-three method

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

11. In a quick sort algorithm, where are smaller elements placed to the pivot during the partition process, assuming we are sorting in increasing

order?

Answer

To the right of the pivot

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

12. What is the main advantage of Quicksort over Merge Sort?

Answer

Quicksort requires less auxiliary space

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

13. Which of the following modifications can help Quicksort perform better on small subarrays?

Answer

Switching to Insertion Sort for small subarrays

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

14. In a quick sort algorithm, what role does the pivot element play?

Answer

It is used to find the largest element in the array

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

15. Which of the following statements is true about the merge sort algorithm?

Answer

It requires additional memory for merging

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

16. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

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

17. What happens during the merge step in Merge Sort?

Answer

Two sorted subarrays are combined into one sorted array

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

18. Which of the following is not true about QuickSort?

Answer

It can be implemented as a stable sort

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

19. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

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

20. Consider the Quick Sort algorithm, which sorts elements in ascending order using the first element as a pivot. Then which of the following input sequences will require the maximum number of comparisons when this algorithm is applied to it?

Answer

22 25 56 67 89

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Nandhini asked her students to arrange a set of numbers in ascending order. She asked the students to arrange the elements using insertion sort, which involves taking each element and placing it in its appropriate position within the sorted portion of the array.

Assist them in the task.

Input Format

The first line of input consists of the value of n, representing the number of array elements.

The second line consists of n elements, separated by a space.

Output Format

The output prints the sorted array, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
67 28 92 37 59
Output: 28 37 59 67 92

Answer

```
#include <stdio.h>


void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Move elements of arr[0..i-1], that are greater than key
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }

        arr[j + 1] = key;
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
}


int main() {
    int n;
```

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

    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John and Mary are collaborating on a project that involves data analysis. They each have a set of age data, one sorted in ascending order and the other in descending order. However, their analysis requires the data to be in ascending order.

Write a program to help them merge the two sets of age data into a single sorted array in ascending order using merge sort.

Input Format

The first line of input consists of an integer N, representing the number of age values in each dataset.

The second line consists of N space-separated integers, representing the ages of participants in John's dataset (in ascending order).

The third line consists of N space-separated integers, representing the ages of participants in Mary's dataset (in descending order).

Output Format

The output prints a single line containing space-separated integers, which represents the merged dataset of ages sorted in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 3 5 7 9
10 8 6 4 2

Output: 1 2 3 4 5 6 7 8 9 10

Answer

```
#include <stdio.h>

// // You are using GCC
void merge(int arr[], int left[], int right[], int left_size, int right_size) {
    int i = 0, j = 0, k = 0;


    while (i < left_size && j < right_size) {
        if (left[i] <= right[j]) {
            arr[k++] = left[i++];
        } else {
            arr[k++] = right[j++];
        }
    }

    while (i < left_size) {
        arr[k++] = left[i++];
    }

    while (j < right_size) {
        arr[k++] = right[j++];
    }
```

```
}

void mergeSort(int arr[], int size) {
    if (size < 2) return;

    int mid = size / 2;

    int left[mid];
    int right[size - mid];

    for (int i = 0; i < mid; i++) {
        left[i] = arr[i];
    }
    for (int i = mid; i < size; i++) {
        right[i - mid] = arr[i];
    }

    mergeSort(left, mid);
    mergeSort(right, size - mid);

    merge(arr, left, right, mid, size - mid);
}

int main() {
    int n, m;
    scanf("%d", &n);
    int arr1[n], arr2[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr1[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr2[i]);
    }
    int merged[n + n];
    mergeSort(arr1, n);
    mergeSort(arr2, n);
    merge(merged, arr1, arr2, n, n);
    for (int i = 0; i < n + n; i++) {
        printf("%d ", merged[i]);
    }
    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are the lead developer of a text-processing application that assists writers in organizing their thoughts. One crucial feature is a character-sorting service that helps users highlight the most critical elements of their text.

To achieve this, you decide to enhance the service to sort characters in descending order using the Quick-Sort algorithm. Implement the algorithm to efficiently rearrange the characters, ensuring that it is sorted in descending order.

Input Format

The first line of the input consists of a positive integer value N, representing the number of characters to be sorted.

The second line of input consists of N space-separated lowercase alphabetical characters.

Output Format

The output displays the set of alphabetical characters, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
a d g j k
Output: k j g d a

Answer

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

#include <stdio.h>

void swap(char *a, char *b) {
    char temp = *a;
    *a = *b;
    *b = temp;
}

int partition(char arr[], int low, int high) {
    char pivot = arr[high]; // pivot
    int i = (low - 1); // Index of smaller element

    for (int j = low; j < high; j++) {
        // If current element is greater than or equal to pivot
        if (arr[j] >= pivot) {
            i++; // increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
```

```
    return (i + 1);
}

void quicksort(char characters[], int low, int high) {
    if (low < high) {
        // pi is partitioning index, arr[pi] is now at right place
        int pi = partition(characters, low, high);

        // Recursively sort elements before partition and after partition
        quicksort(characters, low, pi - 1);
        quicksort(characters, pi + 1, high);
    }
}

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

    char characters[n];

    for (int i = 0; i < n; i++) {
        char input;
        scanf(" %c", &input);
        characters[i] = input;
    }

    quicksort(characters, 0, n - 1);

    for (int i = 0; i < n; i++) {
        printf("%c ", characters[i]);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Kavya, a software developer, is analyzing data trends. She has a list of integers and wants to identify the nth largest number in the list after sorting the array using QuickSort.

To optimize performance, Kavya is required to use QuickSort to sort the list before finding the nth largest number.

Input Format

The first line of input consists of an integer n, representing the size of the array.

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

The third line consists of an integer k, representing the position of the largest

number you need to print after sorting the array.

Output Format

The output prints the k-th largest number in the sorted array (sorted in ascending order).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
-1 0 1 2 -1 -4
3
Output: 0

Answer

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


int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for(int j = low; j < high; j++) {
        if(arr[j] <= pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return i + 1;
}

void quicksort(int arr[], int low, int high) {
    if(low < high) {
```

```
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}

void findNthLargest(int arr[], int n, int k) {
    quicksort(arr, 0, n - 1);
    printf("%d\n", arr[n - k]);
}


int main() {
    int n, k;
    scanf("%d", &n);
    int* nums = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    scanf("%d", &k);
    findNthLargest(nums, n, k);
    free(nums);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Jose has an array of N fractional values, represented as double-point numbers. He needs to sort these fractions in increasing order and seeks your help.

Write a program to help Jose sort the array using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of fractions to be sorted.

The second line consists of N double-point numbers, separated by spaces, representing the fractions array.

Output Format

The output prints N double-point numbers, sorted in increasing order, and rounded to three decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
0.123 0.543 0.321 0.789

Output: 0.123 0.321 0.543 0.789

Answer

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


int compare(double a, double b) {
    if (a < b) return -1;
    else if (a > b) return 1;
    else return 0;
}

void merge(double arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    double L[n1], R[n2];
    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (compare(L[i], R[j]) <= 0)
            arr[k++] = L[i++];
        else
            arr[k++] = R[j++];
    }
```

```
    while (i < n1)
        arr[k++] = L[i++];
    while (j < n2)
        arr[k++] = R[j++];
}

void mergeSort(double arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}


int main() {
    int n;
    scanf("%d", &n);
    double fractions[n];
    for (int i = 0; i < n; i++) {
        scanf("%lf", &fractions[i]);
    }
    mergeSort(fractions, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%.3f ", fractions[i]);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 17

Section 1 : MCQ

1. In division method, if key = 125 and m = 13, what is the hash index?

Answer

8

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

2. Which folding method divides the key into equal parts, reverses some of them, and then adds all parts?

Answer

Folding reversal method

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

3. What is the output of the mid-square method for a key k = 123 if the hash table size is 10 and you extract the middle two digits of k * k?

Answer

2

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

4. Which of the following statements is TRUE regarding the folding method?

Answer

It divides the key into parts and adds them.

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

5. What is the initial position for a key k in a linear probing hash table?

Answer

(k + 1) % table_size

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

6. Which situation causes clustering in linear probing?

Answer

Poor hash function

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

7. In linear probing, if a collision occurs at index i, what is the next index checked?

Answer

(i + 1) % table_size

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

8. Which of the following best describes linear probing in hashing?

Answer

Resolving collisions by linearly searching for the next free slot

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

9. Which C statement is correct for finding the next index in linear probing?

Answer

index = (index + 1) % size;

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

10. In the folding method, what is the primary reason for reversing alternate parts before addition?

Answer

To reduce the chance of collisions caused by similar digit patterns

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

11. What happens if we do not use modular arithmetic in linear probing?

Answer

Index goes out of bounds

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

12. Which of the following values of 'm' is recommended for the division method in hashing?

Answer

A prime number

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

13. What does a deleted slot in linear probing typically contain?

Answer

A special "deleted" marker

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

14. What is the primary disadvantage of linear probing?

Answer

Clustering

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

15. What is the worst-case time complexity for inserting an element in a hash table with linear probing?

Answer

O(n)

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

16. In the division method of hashing, the hash function is typically written as:

Answer

h(k) = k % m

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

17. Which data structure is primarily used in linear probing?

Answer

Array

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

18. What would be the result of folding 123456 into three parts and summing: (12 + 34 + 56)?

Answer

102

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

19. Which of these hashing methods may result in more uniform distribution with small keys?

Answer

Mid-Square

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

20. In C, how do you calculate the mid-square hash index for a key k, assuming we extract two middle digits and the table size is 100?

Answer

((k * k) / 100) % 100

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is building a basic hash table to manage student roll numbers for quick lookup. He decides to use Linear Probing to handle collisions.

Implement a hash table using linear probing where:

The hash function is: index = roll_number % table_sizeOn collision, check subsequent indexes (i+1, i+2, ...) until an empty slot is found.

You need to:

Insert a list of n student roll numbers into the hash table.Print the final state of the hash table. If a slot is empty, print -1.

Input Format

The first line of the input contains two integers n and table_size, where n is the

number of roll numbers to be inserted, and table_size is the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert into the hash table.

Output Format

The output should print a single line with table_size space-separated integers representing the final state of the hash table after all insertions.

If any slot remains unoccupied, it should be represented as -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4 7
50 700 76 85
Output: 700 50 85 -1 -1 -1 76

Answer

```
#include <stdio.h>

#define MAX 100
void initializeTable(int table[], int size)
{

   for (int i = 0; i < size; i++)
{

      table[i] = -1;

}
```

```
}
int linearProbe(int table[], int size, int num)
{

   int index = num % size;
   for (int i = 0; i < size; i++)
{

      int current_index = (index + i) % size;
      if (table[current_index] == -1)
{

         return current_index;

}

}
   return -1;
}
void insertIntoHashTable(int table[], int size, int arr[], int n)
{

   for (int i = 0; i < n; i++)
{

      int roll_number = arr[i];
```

```
        int index_to_insert = linearProbe(table, size, roll_number);
        if (index_to_insert != -1)
{

            table[index_to_insert] = roll_number;


}


}
}
void printTable(int table[], int size)
{

    for (int i = 0; i < size; i++)
{

        printf("%d", table[i]);
        if (i < size - 1)
{

            printf(" ");


}


}
    printf("\n");
}
```

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

    int arr[MAX];
    int table[MAX];

    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
    printTable(table, table_size);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Priya is developing a simple student management system. She wants to store roll numbers in a hash table using Linear Probing, and later search for specific roll numbers to check if they exist.

Implement a hash table using linear probing with the following operations:

Insert all roll numbers into the hash table.For a list of query roll numbers, print "Value x: Found" or "Value x: Not Found" depending on whether it exists in the table.

Input Format

The first line contains two integers, n and table_size — the number of roll numbers to insert and the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert.

The third line contains an integer q — the number of queries.

The fourth line contains q space-separated integers — the roll numbers to search for.

Output Format

The output print q lines — for each query value x, print: "Value x: Found" or "Value x: Not Found"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 10
21 31 41 51 61
3
31 60 51
Output: Value 31: Found
Value 60: Not Found
Value 51: Found

Answer

```
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size)

{

   for (int i = 0; i < size; i++)

{

      table[i] = -1;
```

```
}

}

int findInsertIndex(int table[], int size, int num)

{


   int initial_index = num % size;
   for (int i = 0; i < size; i++)

{


      int current_index = (initial_index + i) % size;
      if (table[current_index] == -1)

{


         return current_index;


}


}
   return -1;

}

void insertIntoHashTable(int table[], int size, int arr[], int n)

{


   for (int i = 0; i < n; i++)
{
```

```
        int roll_number = arr[i];
        int index_to_insert = findInsertIndex(table, size, roll_number);
        if (index_to_insert != -1)
{

            table[index_to_insert] = roll_number;

}
}
}
int searchInHashTable(int table[], int size, int num)
{

    int initial_index = num % size;
    for (int i = 0; i < size; i++)
{

        int current_index = (initial_index + i) % size;
        if (table[current_index] == num)
{

            return 1;

}
        if (table[current_index] == -1)
```

```
{

        return 0;


}


}
    return 0;

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

    int arr[MAX], table[MAX];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);

    int q, x;
    scanf("%d", &q);
    for (int i = 0; i < q; i++) {
        scanf("%d", &x);
        if (searchInHashTable(table, table_size, x))
            printf("Value %d: Found\n", x);
        else
            printf("Value %d: Not Found\n", x);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a messaging application, users maintain a contact list with names and corresponding phone numbers. Develop a program to manage this contact list using a dictionary implemented with hashing.

The program allows users to add contacts, delete contacts, and check if a specific contact exists. Additionally, it provides an option to print the contact list in the order of insertion.

Input Format

The first line consists of an integer n, representing the number of contact pairs to be inserted.

Each of the next n lines consists of two strings separated by a space: the name of the contact (key) and the corresponding phone number (value).

The last line contains a string k, representing the contact to be checked or removed.

Output Format

If the given contact exists in the dictionary:

1. The first line prints "The given key is removed!" after removing it.
2. The next n - 1 lines print the updated contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

If the given contact does not exist in the dictionary:

1. The first line prints "The given key is not found!".
2. The next n lines print the original contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
Alice 1234567890
Bob 9876543210
Charlie 4567890123
Bob

Output: The given key is removed!
Key: Alice; Value: 1234567890
Key: Charlie; Value: 4567890123

Answer

```
// You are using GCC
#include <stdio.h>
#include <string.h>

#define MAX 50
```

```
typedef struct {
    char name[11];
    char phone[11];
    int isActive;
} Contact;

int findContact(Contact contacts[], int n, char key[]) {
    for (int i = 0; i < n; i++) {
        if (contacts[i].isActive && strcmp(contacts[i].name, key) == 0) {
            return i;
        }
    }
    return -1;
}

int main() {
    int n;
    Contact contacts[MAX];
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%s %s", contacts[i].name, contacts[i].phone);
        contacts[i].isActive = 1;
    }

    char key[11];
    scanf("%s", key);

    int index = findContact(contacts, n, key);

    if (index != -1) {
        contacts[index].isActive = 0;
        printf("The given key is removed!\n");
    } else {
        printf("The given key is not found!\n");
    }

    for (int i = 0; i < n; i++) {
        if (contacts[i].isActive) {
            printf("Key: %s; Value: %s\n", contacts[i].name, contacts[i].phone);
        }
    }
```

```
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Develop a program using hashing to manage a fruit contest where each fruit is assigned a unique name and a corresponding score. The program should allow the organizer to input the number of fruits and their names with scores.

Then, it should enable them to check if a specific fruit, identified by its name, is part of the contest. If the fruit is registered, the program should display its score; otherwise, it should indicate that it is not included in the contest.

Input Format

The first line consists of an integer N, representing the number of fruits in the contest.

The following N lines contain a string K and an integer V, separated by a space, representing the name and score of each fruit in the contest.

The last line consists of a string T, representing the name of the fruit to search for.

Output Format

If T exists in the dictionary, print "Key "T" exists in the dictionary.".

If T does not exist in the dictionary, print "Key "T" does not exist in the dictionary.".

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 2
banana 2
apple 1
Banana
Output: Key "Banana" does not exist in the dictionary.

Answer

```
// You are using GCC
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define SIZE 20

typedef struct {
    char name[21];
    int score;
    int isOccupied;
} Fruit;

int hash(char *key) {
    int hashVal = 0;
    for (int i = 0; key[i] != '\0'; i++) {
```

```
        hashVal = (hashVal * 31 + key[i]) % SIZE;
    }
    return hashVal;
}

void insert(Fruit table[], char *key, int value) {
    int idx = hash(key);
    while (table[idx].isOccupied) {
        idx = (idx + 1) % SIZE;
    }
    strcpy(table[idx].name, key);
    table[idx].score = value;
    table[idx].isOccupied = 1;
}

int search(Fruit table[], char *key, int *value) {
    int idx = hash(key);
    int startIdx = idx;
    while (table[idx].isOccupied) {
        if (strcmp(table[idx].name, key) == 0) {
            *value = table[idx].score;
            return 1;
        }
        idx = (idx + 1) % SIZE;
        if (idx == startIdx) break;
    }
    return 0;
}

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

    Fruit table[SIZE] = {0};

    for (int i = 0; i < N; i++) {
        char name[21];
        int score;
        scanf("%s %d", name, &score);
        insert(table, name, score);
    }
```

```
    char searchKey[21];
    scanf("%s", searchKey);

    int foundScore;
    if (search(table, searchKey, &foundScore)) {
        printf("Key \"%s\" exists in the dictionary.\n", searchKey);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", searchKey);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.
Email: 241501020@rajalakshmi.edu.in
Roll no: 241501020
Phone: 9025837734
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are provided with a collection of numbers, each represented by an array of integers. However, there's a unique scenario: within this array, one element occurs an odd number of times, while all other elements occur an even number of times. Your objective is to identify and return the element that occurs an odd number of times in this arrangement.

Utilize mid-square hashing by squaring elements and extracting middle digits for hash codes. Implement a hash table for efficient integer occurrence tracking.

Note: Hash function: squared = key * key.

Example

Input:

7

2 2 3 3 4 4 5

Output:

5

Explanation

The hash function and the calculated hash indices for each element are as follows:

2 -> hash(2*2) % 100 = 4

3 -> hash(3*3) % 100 = 9

4 -> hash(4*4) % 100 = 16

5 -> hash(5*5) % 100 = 25

The hash table records the occurrence of each element's hash index:

Index 4: 2 occurrences

Index 9: 2 occurrences

Index 16: 2 occurrences

Index 25: 1 occurrence

Among the elements, the integer 5 occurs an odd number of times (1 occurrence) and satisfies the condition of the problem. Therefore, the program outputs 5.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer representing the element that occurs an odd

number of times.

If no such element exists, print -1.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
2 2 3 3 4 4 5

Output: 5

Answer

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

#define MAX_SIZE 100

unsigned int hash(int key, int tableSize)

{


    unsigned int square = key * key;
    unsigned int mid = (square / 10) % 100;
    return mid % tableSize;

}

int getOddOccurrence(int arr[], int size)

{


    int tableSize = MAX_SIZE;
    int hashTable[MAX_SIZE];
    int keyTable[MAX_SIZE];
    int i;
```

```
    for (i = 0; i < tableSize; i++)
    {

        hashTable[i] = 0;
        keyTable[i] = 0;


    }

       for (i = 0; i < size; i++)
    {

        int key = arr[i];
        int idx = hash(key, tableSize);

        while (hashTable[idx] != 0 && keyTable[idx] != key)
    {

            idx = (idx + 1) % tableSize;

    }

        if (hashTable[idx] == 0)
    {

            keyTable[idx] = key;
            hashTable[idx] = 1;

    } else
    {
```

```
            hashTable[idx]++;

    }

    }
        for (i = 0; i < tableSize; i++)
    {
            if (hashTable[i] % 2 == 1 && hashTable[i] != 0)
    {

                return keyTable[i];

    }

    }
        return -1;
    }
    int main() {
        int n;
        scanf("%d", &n);

        int arr[MAX_SIZE];
        for (int i = 0; i < n; i++) {
            scanf("%d", &arr[i]);
        }
        printf("%d\n", getOddOccurrence(arr, n));
```

```
    return 0;
}
```

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