RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

Affiliated to Anna University, Chennai,

Rajalakshmi Nagar, Thandalam – 602 105

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS23231- DATA STRUCTURES
Laboratory Record Note Book

Name : ……

Register No. : ……..

Year / Branch / Section : ……...

Semester : ……..

AcademicYear: ……...

Scanned with OKEN Scanner

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

Affiliated to Anna University, Chennai,

Rajalakshmi Nagar, Thandalam – 602 105

BONAFIDE CERTIFICATE

Name: ……

Academic Year: …… **Semester:** …… **Branch:** ……

Register No.

Certified that this is the bonafide record of work done by the above student in the*..... *Laboratory during the academic year 2023- 2024

Signature of Faculty in-charge

Submitted for the Practical Examination held on……

Internal Examiner **External Examiner**

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Arrays and Functions

Attempt : 1
Total Mark : 5
Marks Obtained : 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.

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

4500

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

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

Answer

```
#include<stdio.h>
int main()
{
    int n,i,sum=0;;
    scanf("%d",&n);
    int salary[1000];
    for(i=0;i<n;i++)
    {

        scanf("%d",&salary[i]);
        printf("Employee %d: %d\n",i+1,salary[i]);
        sum+=salary[i];

    }
    int max_salary=salary[0],min_salary=salary[0];
    for(i=0;i<n;i++)
    {
        if(salary[i]>max_salary)
        {
            max_salary=salary[i];
        }
        if(salary[i]<min_salary)
        {
            min_salary=salary[i];
        }
    }
    float average_salary= (float)sum/n;
    printf("Average Salary: %.2f\n",average_salary);
    printf("Highest Salary: %d\n",max_salary);
    printf("Lowest Salary: %d\n",min_salary);
```

Scanned with OKEN Scanner

}

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

2. 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

Scanned with OKEN Scanner

```
#include <stdio.h>

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

3. Problem Statement

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

Answer

```
#include<stdio.h>
int main()
{
    int n;
    int m;
    scanf("%d\n%d\n",&n,&m);
    int matrix[n][m];
    matrix[5][100];
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            scanf("%d",&matrix[i][j]);
        }
    }
    if(matrix[n-1][m-1]%2==0)
    {
        printf("%d is even",matrix[n-1][m-1]);
    }
    else
    printf("%d is odd",matrix[n-1][m-1]);
}
```

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

4. 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

Scanned with OKEN Scanner

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)."

240701244

Scanned with OKEN Scanner

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

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n;
    int arr[10000];
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    int small=arr[0],large=arr[0],sum=0;
    for(int i=0;i<n;i++)
    {
        sum+=arr[i];
        if(arr[i]>large)
        large=arr[i];
        else if(arr[i]<small)
        small=arr[i];
    }
    double avg=(double)sum/n;
    int choice[5];
    for (int i=1;i<=5;i++)
    {
        scanf("%d\n",&choice[i]);
        switch(choice[i])
        {
            case 1:
            printf("The smallest number is: %d\n",small);
            break;
            case 2:
```

Scanned with OKEN Scanner

```
        printf("The largest number is: %d\n",large);
        break;
        case 3:
        printf("The sum of the numbers is: %d\n",sum);
        break;
        case 4:
        printf("The average of the numbers is: %.2f\n",avg);
        break;
        case 5:
        printf("Exiting the program\n");
        exit(0);
        default:
        printf("Invalid choice!Please enter a valid option (1-5).\n");
    }
}

}
```

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

5. 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.

Scanned with OKEN Scanner

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Output: 6 15 24

Answer

```
#include <stdio.h>

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

Scanned with OKEN Scanner

Status : Correct

Marks : 1/1

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Pointers

Attempt : 1
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. Problem Statement

Rajwinder wants a program to determine retirement details for a person based on their age.

Create a program that uses a structure called Person to hold the age as an attribute with a pointer.

If the age is under 18, display "Invalid".If the age is 65 or older, print "Already retired!".Otherwise, calculate and output the retirement year, remaining years, and remaining days until retirement.

Note: Age 65 is considered as retirement age. Assume the current year as 2023 and there are 365 days per year for calculation.

Input Format

Scanned with OKEN Scanner

The input consists of an integer representing the person's age.

Output Format

If the age is under 18, the output displays "Invalid" and terminates.

If the age is 65 or older, the output displays "Already retired!" and terminates.

Otherwise, the output displays the following.

1. The first line displays "Retirement Year: " followed by an integer representing the retirement year.
2. The second line displays "Remaining Years: " followed by an integer representing the remaining years left for retirement.
3. The third line displays "Remaining Days: " followed by an integer representing the remaining days left for retirement.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 43

Output: Retirement Year: 2045
Remaining Years: 22
Remaining Days: 8030

Answer

```
#include<stdio.h>
typedef struct{
    int age;
}Person;
void det(Person*P){
    int cy=2023;
    int rage=65;
    if(P->age<18){
        printf("Invalid");
        return;
    }
    if(P->age>=rage){
        printf("Already retired!");
        return;
```

Scanned with OKEN Scanner

```
    }
    int ryears=rage-P->age;
    int rdays=ryears*365;
    int ryear=cy+ryears;
    printf("Retirement Year: %d\n",ryear);
    printf("Remaining Years: %d\n",ryears);
    printf("Remaining Days: %d\n",rdays);

}
int main(){
    Person P;
    scanf("%d",&P.age);
    det(&P);
    return 0;
}
```

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

2. Problem Statement

Sam is developing a program for analyzing daily temperature fluctuations. Users input the number of days, followed by daily temperature values whose memory is allocated using malloc.

The program calculates and displays the following:

The absolute temperature changes between consecutive days (The first value remains the same).The average temperature of adjacent days.

This allows users to gain insights into daily temperature variations for better analysis.

For Example,

Let us assume the temperature for 3 days as 25.5, 28.0, and 23.5.

The absolute differences:

Day 1: (N/A, as there is no previous day) = 25.50Day 2: abs(28.0 - 25.5) = 2.50Day 2: abs(23.5 - 28.0) = 4.50

The average temperatures:

Scanned with OKEN Scanner

Day 1: (N/A, as there is no previous day) = 25.50Day 2: (25.5 + 23.5) / 2.0 = 24.50Day 3: (N/A, as there is no next day) = 23.50

Input Format

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

The second line consists of N space-separated float values, representing the temperature values for N days.

Output Format

The first line displays the absolute temperature change for N days as float values, rounded to two decimal places, separated by a space.

The second line displays the average temperature for N days as float values, rounded to two decimal places, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
25.5 28.0 23.5

Output: 25.50 2.50 4.50
25.50 24.50 23.50

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
    int N;
    scanf("%d",&N);
    float *temp;
    temp=(float*)malloc(N*sizeof(float));
    for(int i=0;i<N;i++)
    {
        scanf("%f",&temp[i]);
```

240701244

Scanned with OKEN Scanner

```
    }
    printf("%.2f",temp[0]);
    for(int i=1;i<N;i++)
    printf("%.2f",fabs(temp[i]-temp[i-1]));
    printf("\n");
    printf("%.2f",temp[0]);
    for(int i=1;i<N-1;i++)
    {
        printf("%.2f",(temp[i-1]+temp[i+1])/2.0);
        temp[i]=(temp[i-1]+temp[i+1])/2.0;

    }
    printf("%.2f",temp[N-1]);
    free(temp);

}
```

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

3. Problem Statement

Daniel is working on a project that involves analyzing data stored in float arrays. He needs to determine whether a given float array contains only positive numbers.

To achieve this, he needs a program that can accurately evaluate the contents of float arrays using malloc().

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 float values, representing the elements of the array.

Output Format

If all the array elements are positive, print "All elements are positive."

If the array contains at least one positive element, print "At least one element is positive."

240701244

Scanned with OKEN Scanner

If there are no positive elements in the array, print "No positive elements in the array."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
50.0 -2.3 3.7 -4.8 5.2
Output: At least one element is positive.

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int N;
    scanf("%d",&N);
    float *arr;
    arr=(float*)malloc(N*sizeof(arr));
    int allpositive=1;
    int atleastpositive=0;
    for(int i=0;i<N;i++)
    {
        scanf("%f",&arr[i]);
    }
    for(int i=0;i<N;i++)
    {
        if(arr[i]>0)
        {
            atleastpositive=1;
        }
        else
        {
            allpositive=0;
        }
    }
    if(allpositive)
    printf("All elements are positive.");
```

Scanned with OKEN Scanner

```
    else if(atleastpositive)
    printf("At least one element is positive.");
    else
    printf("No positive elements in the array.");
    free(arr);
}
```

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

4. Problem Statement

Raj wants to create a program using pointers and a structure named Employee to manage employee information.

He seeks your assistance to input the employee's name, salary, and hours worked. Implement a salary increase based on hours worked, and calculate the final salary. Calculate the total salary for 30 days. Display the results of the final and total salary.

Salary increase criteria:

If hours worked >= 12, the increase is Rs. 150.00.If hours worked >= 10, but less than 12, the increase is Rs. 100.00.If hours worked >= 8, but less than 10, the increase is Rs. 50.00.If hours worked < 8, there is no increase.

Input Format

The first line of input consists of a string, representing the Employee's name.

The second line consists of a double-point number, representing the Employee's current salary.

The third line consists of an integer, representing the number of hours worked by the employee.

Output Format

The first line of output prints "Final Salary: Rs. " followed by a double value, representing the final salary, rounded off to two decimal places.

The second line prints "Total Salary: Rs. " followed by a double value, representing the total salary for 30 days, rounded off to two decimal places.

Scanned with OKEN Scanner

Refer to the sample outputs for formatting specifications.

Sample Test Case

Input: Akil
3000.00
6

Output: Final Salary: Rs. 3000.00
Total Salary: Rs. 90000.00

Answer

```
#include<stdio.h>
struct Employee
{
   char name[50];
   double salary;
   int hours;
};
int main()
{
   struct Employee emp;
   scanf("%s",emp.name);
   scanf("%lf\n%d",&emp.salary,&emp.hours);
   if(emp.hours >=12)
   {
      emp.salary+=150.00;
   }
   else if(emp.hours >=10 && emp.hours<12)
   {
      emp.salary+=100.00;
   }
   else if(emp.hours >=8 && emp.hours<10)
   {
      emp.salary+=50.00;
   }
   printf("Final Salary: Rs.%.2lf\n",emp.salary);
   printf("Total Salary: Rs.%.2lf",emp.salary*30);
}
```

Scanned with OKEN Scanner

Status : Correct *Marks : 1/1*

5. Problem Statement

Ria is a mathematician who loves exploring combinatorics. She is working on a project that involves calculating permutations.

Ria wants to create a program that takes the values of n and r as input and calculates the permutations of n elements taken r at a time.

Write a program using pointers and a function calculatePermutations that, given the values of n and r, calculates and prints the permutations of n elements taken r at a time.

Permutation: n! / (n - r)!

Input Format

The first line consists of an integer n, representing the total number of elements.

The second line consists of an integer r, representing the number of elements to be taken at a time.

Output Format

The output prints the result of the permutation.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3
Output: 24

Answer

```
#include<stdio.h>
long fact(int n,int r)
{
```

Scanned with OKEN Scanner

```
    long f=1;
    for(int i=0;i<r;i++)
    f*=(n-i);
    return f;
}
int main()
{
    int n,r;
    scanf("%d",&n);
    scanf("%d",&r);
    long x=fact(n,r);
    printf("%ld\n",x);
    return 0;
}
```

Status : Correct

Marks : 1/1

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : MCQ

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

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

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

void rearrange (struct node* list) {
   struct node *p,q;
   int temp;
   if (! List || ! list->next) return;
```

Scanned with OKEN Scanner

```
    p=list; q=list->next;
    while(q) {
        temp=p->value; p->value=q->value;
        q->value=temp;p=q->next;
        q=p?p->next:0;
    }
}
```

Answer

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

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

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

What will be the final linked list after the deletion?

Answer

15 -> 16 -> 6

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

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

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

```
struct node {
    int data;
    struct node* next;
};
static void reverse(struct node** head_ref) {
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL) {
```

Scanned with OKEN Scanner

```
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}
```

Answer

*head_ref = prev;

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

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

Answer

It stores the last element of the list

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

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

What will be the final linked list after the deletion?

Answer

13 -> 16 -> 22 -> 45 -> 16

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

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

Answer

5 10 15 20 25

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

Scanned with OKEN Scanner

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

Answer

Possible if X is not last node.

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

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

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

Answer

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

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

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

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

I and III

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

Scanned with OKEN Scanner

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

Answer

Binary search

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 2
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.

Scanned with OKEN Scanner

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

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

Output Format

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

Sample Test Case

Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0
Output: 18

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
   int co;
   int ex;
   struct Node*next;
}node;
void insertnode(node**head,int co,int ex)
{
   node*newnode=(node*)malloc(sizeof(node));
   newnode->co=co;
   newnode->ex=ex;
   newnode->next=NULL;
   if(*head==NULL||(*head)->ex<ex)
   {
      newnode->next=*head;
      *head=newnode;
   }
   else
```

Scanned with OKEN Scanner

```
{
    node*temp=*head;
    while(temp->next!=NULL && temp->next->ex>ex)
    {
        temp=temp->next;
    }
    if(temp->next!=NULL&&temp->next->ex==ex)
    {
        temp->next->co+=co;
    }
    else
    {
        newnode->next=temp->next;
        temp->next=newnode;
    }
  }
}
node*addpoly(node*poly1,node*poly2)
{
    node*result=NULL;
    while(poly1!=NULL)
    {
        insertnode(&result,poly1->co,poly1->ex);
        poly1=poly1->next;
    }
    while(poly2!=NULL)
    {
        insertnode(&result,poly2->co,poly2->ex);
        poly2=poly2->next;
    }
    return result;
}
int sumofpoly(node*result)
{
    int sum=0;
    node*temp=result;
    while(temp!=NULL)
    {
        sum+=temp->co;
        temp=temp->next;
    }
    return sum;
```

240701244

Scanned with OKEN Scanner

```
}
int main()
{
    int n,m;
    node*poly1=NULL;
    node*poly2=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        int co,ex;
        scanf("%d %d",&co,&ex);
        insertnode(&poly1,co,ex);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++)
    {
        int co,ex;
        scanf("%d %d",&co,&ex);
        insertnode(&poly2,co,ex);
    }
    node*result=addpoly(poly1,poly2);
    int resultsum=sumofpoly(result);
    printf("%d",resultsum);


}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 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.

Scanned with OKEN Scanner

Position starts from 1.

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2

Output: 8 3 1 7

Answer

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

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

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

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

Scanned with OKEN Scanner

```
        tail->next=newnode;
        tail=newnode;
    }
}
void display_List(){
    struct node*temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
}
void deleteNode(int pos){
    if(pos<1){
        printf("Invalid position. Deletion not possible.");
        return;
    }

    struct node*temp=head;
    if(pos==1){
        if(head!=NULL)
    {
        head=head->next;
        free(temp);
    }
    display_List();
    return;
    }
    struct node*prev=NULL;
    int count=1;
    while(temp!=NULL&&count<pos){
        prev=temp;
        temp=temp->next;
        count++;
    }
    if(temp==NULL){
        printf("Invalid position. Deletion not possible.");
        return;
    }
    prev->next=temp->next;
    if(temp==tail){
        tail=prev;
    }
```

240701244

Scanned with OKEN Scanner

```
free(temp);
display_List();
}

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

    scanf("%d", &num_elements);

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

    scanf("%d", &pos_to_delete);

    deleteNode(pos_to_delete);

    return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are working on a text processing tool and need to implement a feature that allows users to insert characters at a specific position.

Implement a program that takes user inputs to create a singly linked list of characters and inserts a new character after a given index in the list.

Input Format

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

The second line consists of a sequence of N characters, representing the linked list.

The third line consists of an integer index, representing the index(0-based) after

Scanned with OKEN Scanner

which the new character node needs to be inserted.

The fourth line consists of a character value representing the character to be inserted after the given index.

Output Format

If the provided index is out of bounds (larger than the list size):

1. The first line of output prints "Invalid index".
2. The second line prints "Updated list: " followed by the unchanged linked list values.

Otherwise, the output prints "Updated list: " followed by the updated linked list after inserting the new character after the given index.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
a b c d e
2
X
Output: Updated list: a b c X d e

Answer

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
   char data;
   struct Node* next;
};
struct Node*createNode(char data){
   struct Node*newnode=(struct Node*)malloc(sizeof(struct Node));
   newnode->data=data;
   newnode->next=NULL;
   return newnode;
}
```

Scanned with OKEN Scanner

```
void printList(struct Node*head){
    struct Node*temp=head;
    while(temp !=NULL)
    {
        printf("%c ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
void insertAfter(struct Node*head,int index,char value){
    struct Node*temp=head;
    int count=0;
    while(temp!=NULL&&count<index)
    {
        temp=temp->next;
        count++;
    }
    if(temp==NULL){
        printf("Invalid index\n");
        printf("Updated list:");
        printList(head);
        return;
    }
    struct Node*newnode=createNode(value);
    newnode->next=temp->next;
    temp->next=newnode;
    printf("Updated list:");
    printList(head);
}
int main(){
    int N,index;
    char value;
    scanf("%d",&N);
    struct Node*head=NULL,*tail=NULL;
    for(int i=0;i<N;i++)
    {
        char ch;
        scanf(" %c",&ch);
        struct Node*newnode=createNode(ch);
        if(head==NULL){
            head=newnode;
            tail=newnode;
```

240701244

Scanned with OKEN Scanner

```
        }
        else{
            tail->next=newnode;
            tail=newnode;
        }
    }
    scanf("%d",&index);
    scanf(" %c",&value);
    insertAfter(head,index,value);
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

The second line consists of N space-separated integers.

Output Format

Scanned with OKEN Scanner

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;
};

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;
```

Scanned with OKEN Scanner

```
    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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

Input Format

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

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

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

Scanned with OKEN Scanner

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3.8
3.2
3.5
4.1
2
Output: GPA: 4.1
GPA: 3.2
GPA: 3.8

Answer

```
#include<stdio.h>
#include<stdlib.h>
struct Node{
   float GPA;
   struct Node*next;
};
struct Node* createNode(float GPA){
   struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
   newNode->GPA=GPA;
   newNode->next=NULL;
   return newNode;
}
void insertFront(struct Node** head,float GPA){
   struct Node* newNode=createNode(GPA);
   newNode->next=*head;
   *head=newNode;
}
void deleteAtPosition(struct Node** head,int position){
   if(*head==NULL){
      return;
```

240701244

Scanned with OKEN Scanner

```
}
struct Node* temp=*head;
if(position==1){
    *head=temp->next;
    free(temp);
    return;
}
for(int i=1;temp!=NULL && i<position-1;i++){
    temp=temp->next;
}
if(temp==NULL||temp->next==NULL){
    return;
}
struct Node* nodeToDelete=temp->next;
temp->next=temp->next->next;
free(nodeToDelete);
}
void displayList(struct Node*head){
    struct Node*temp=head;
    while(temp!=NULL){
        printf("GPA: %.1f\n",temp->GPA);
        temp=temp->next;
    }
}
int main()
{
    int n,position;
    scanf("%d",&n);
    struct Node*head=NULL;
    for(int i=0;i<n;i++){
        float GPA;
        scanf("%f",&GPA);
        insertFront(&head,GPA);
    }
    scanf("%d",&position);
    deleteAtPosition(&head,position);
    displayList(head);
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Output Format

Scanned with OKEN Scanner

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 85 47 62 31

Output: 23 85 47 62 31

Answer

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

Scanned with OKEN Scanner

```
    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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

Scanned with OKEN Scanner

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

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 50 40 30 20 10
Middle Element: 30

Answer

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

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

Scanned with OKEN Scanner

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

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

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

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

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


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


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

    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 2
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

John is working on a math processing application, and his task is to simplify polynomials entered by users. The polynomial is represented as a linked list, where each node contains two properties:

Coefficient of the term.

Exponent of the term.

John's goal is to combine all the terms that have the same exponent, effectively simplifying the polynomial.

Input Format

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

Scanned with OKEN Scanner

The next n lines of input consist of two integers, representing the coefficient and exponent of the polynomial in each line separated by space.

Output Format

The first line of output prints the original polynomial in the format 'cx^e + cx^e + ...' (where c is the coefficient and e is the exponent of each term).

The second line of output displays the simplified polynomial in the same format as the original polynomial.

If the polynomial is 0, then only '0' will be printed.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 2
3 1
6 2
Output: Original polynomial: 5x^2 + 3x^1 + 6x^2
Simplified polynomial: 11x^2 + 3x^1

Answer

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

Scanned with OKEN Scanner

```
struct polynomial{
    struct Node*head;
};
void initpolynomial(struct polynomial*polynomial){
    polynomial->head=NULL;
}
void add(struct polynomial*polynomial,int coeff,int exp){
    struct Node*newNode=createNode(coeff,exp);
    if(polynomial->head==NULL){
        polynomial->head=newNode;
    }else{
        struct Node*ptr=polynomial->head;
        while(ptr->next!=NULL){
            ptr=ptr->next;
        }
        ptr->next=newNode;
    }
}
void display(struct polynomial*polynomial){
    if(polynomial->head==NULL){
        printf("0");
        return;
    }
    struct Node*current=polynomial->head;
    while(current!=NULL){
        printf("%dx^%d",current->coeff,current->exp);
        if(current->next!=NULL){
            printf(" + ");
        }
        current=current->next;
    }
}
void simp(struct polynomial*polynomial){
    if(polynomial->head==NULL||polynomial->head->next==NULL){
        return;
    }
    struct Node*ptr1=polynomial->head;
    struct Node*ptr2;
    struct Node*prev=NULL;
    while(ptr1!=NULL){
        ptr2=ptr1->next;
        prev=ptr1;
```

240701244

Scanned with OKEN Scanner

```
        while(ptr2!=NULL){
            if(ptr1->exp==ptr2->exp){
                ptr1->coeff+=ptr2->coeff;
                prev->next=ptr2->next;
                free(ptr2);
                ptr2=prev->next;
            }else{
                prev=ptr2;
                ptr2=ptr2->next;
            }
        }
        ptr1=ptr1->next;
    }
    }
    int main()
    {
        struct polynomial polynomial;
        initpolynomial(&polynomial);
        int n;
        scanf("%d",&n);

        for(int i=0;i<n;i++){
            int coeff,exp;
            scanf("%d %d",&coeff,&exp);
            add(&polynomial,coeff,exp);
        }
        printf("Original polynomial: ");
        display(&polynomial);

        printf("\nSimplified polynomial: ");
        simp(&polynomial);
        display(&polynomial);

        return 0;
    }
```

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

2. Problem Statement

Keerthi is a tech enthusiast and is fascinated by polynomial expressions.

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

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_term 50
typedef struct
{
    int coeff;
    int exp;
}term;
void sort_poly(term poly[],int size)
{
    for(int i=0;i<size-1;i++){
        for(int j=i+1;j<size;j++){
            if(poly[i].exp<poly[j].exp){
                term temp=poly[i];
                poly[i]=poly[j];
                poly[j]=temp;
            }
        }
    }
}
void multiply(term poly1[],int n,term poly2[],int m,term result[],int *res_size){
    *res_size=0;
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            int new_coeff=poly1[i].coeff*poly2[j].coeff;
            int new_exp=poly1[i].exp+poly2[j].exp;
            int found=0;
            for(int k=0;k<*res_size;k++)
            {
                if(result[k].exp==new_exp){
                    result[k].coeff+=new_coeff;
                    found=1;
```

Scanned with OKEN Scanner

```
                break;
            }
        }
        if(!found){
            result[*res_size].coeff=new_coeff;
            result[*res_size].exp=new_exp;
            (*res_size)++;
        }
    }
  }
}
void delete_term(term poly[],int *size,int exp_to_delete){
    int new_size=0;
    for(int i=0;i<*size;i++){
        if(poly[i].exp!=exp_to_delete){
            poly[new_size++]=poly[i];
        }
    }
    *size=new_size;
}
void display(term poly[],int size)
{
    if(size==0){
        printf("0\n");
        return;
    }
    for(int i=0;i<size;i++){
        if(i>0)printf(" + ");
        if(poly[i].exp==0){
            printf("%d",poly[i].coeff);

        }else if (poly[i].exp==1){
            printf("%dx",poly[i].coeff);
        }else{
            printf("%dx^%d",poly[i].coeff,poly[i].exp);
        }

    }
    printf("\n");
}
int main()
{
```

240701244

Scanned with OKEN Scanner

```
    int n,m,del_exp;
    scanf("%d",&n);
    term poly1[n];
    for(int i=0;i<n;i++){
        scanf("%d %d",&poly1[i].coeff,&poly1[i].exp);
    }
    scanf("%d",&m);
    term poly2[n];
    for(int i=0;i<m;i++){
        scanf("%d %d",&poly2[i].coeff,&poly2[i].exp);
    }
    term result[50];
    int res_size;
    multiply(poly1,n,poly2,m,result,&res_size);
    printf("Result of the multiplication: ");
    display(result, res_size);
    scanf("%d",&del_exp);
    delete_term(result,&res_size,del_exp);
    printf("Result after deleting the term: ");
    display(result,res_size);
    return 0;

}
```

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

3. 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

240701244

Scanned with OKEN Scanner

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>
struct node{
    int coeff,power;
```

240701244

Scanned with OKEN Scanner

```
    struct node*next;
};
struct node*insert(struct node* start,int coeff,int power){
    struct node*newnode=(struct node*)malloc(sizeof(struct node));
    newnode->coeff=coeff;
    newnode->power=power;
    newnode->next=NULL;
    if(start ==NULL||start->power>power){
        newnode->next=start;
        return newnode;
    }
    struct node*ptr=start;
    while(ptr->next!=NULL&&ptr->next->power<power)
        ptr=ptr->next;
    newnode->next=ptr->next;
    ptr->next=newnode;
    return start;
}
void print(struct node*start){
    if(start==NULL){
        printf("0\n");
        return;
    }
    while(start!=NULL){
        printf("%dx^%d ",start->coeff,start->power);
        if(start->next!=NULL)
            printf(" + ");
        start=start->next;

    }
    printf("\n");
}
struct node*polyadd(struct node*poly1,struct node*poly2)
{
    struct node*result=NULL;
    struct node*p1=poly1,*p2=poly2;
    while(p1!=NULL||p2!=NULL)
    {
        int coeff,power;
        if(p2==NULL||(p1!=NULL&&p1->power<p2->power)){
            coeff=p1->coeff;
            power=p1->power;
```

240701244

Scanned with OKEN Scanner

```
        p1=p1->next;
    }
    else if(p1==NULL||(p2!=NULL && p2->power<p1->power)){
        coeff=p2->coeff;
        power=p2->power;
        p2=p2->next;
    }
    else{
        coeff=p1->coeff+p2->coeff;
        power=p1->power;
        p1=p1->next;
        p2=p2->next;
    }
    if(coeff!=0)
        result=insert(result,coeff,power);
    }
    return result;

}
int main()
{
    struct node*poly1=NULL,*poly2=NULL;
    int coeff,power;
    while(1){
        scanf("%d %d",&coeff,&power);
        if(coeff==0&&power==0)
        break;
        poly1=insert(poly1,coeff,power);
    }
    while(1){
        scanf("%d %d",&coeff,&power);
        if(coeff==0&&power==0)
        break;
        poly2=insert(poly2,coeff,power);

    }
    print(poly1);
    print(poly2);
    struct node*result=polyadd(poly1,poly2);
    print(result);
    return 0;
}
```

240701244

Scanned with OKEN Scanner

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_PAH_modified

Attempt : 2
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. 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:

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
    int data;
    struct Node*next;
}node;
node*create(int e)
{
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=e;
    newnode->next=NULL;
    return newnode;
}
void printlist(node*head){
    node*temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
}
int main(){
    int n,e;
    node*even=NULL;
    node*odd=NULL;
    node*oddtail=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        node*newnode=create(e);
        if(e%2==0){
            newnode->next=even;
            even=newnode;
        }
        else{
            if(odd==NULL){
                odd=newnode;
                oddtail=newnode;
            }
            else{
                oddtail->next=newnode;
                oddtail=newnode;
```

240701244

Scanned with OKEN Scanner

```
        }
    }
}
if(even==NULL){
    even=odd;
}
else{
    node*temp=even;
    while(temp->next!=NULL)
    {
        temp=temp->next;
    }
    temp->next=odd;
    }
printlist(even);
}
```

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

2. 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.

240701244

Scanned with OKEN Scanner

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 data;
   struct Node*next;
}node;
void insert(node**head,int e)
{
   node*newnode=(node*)malloc(sizeof(node));
   newnode->data=e;
   newnode->next=NULL;
   if(*head==NULL){
      *head=newnode;
   }
   else
   {
      node*temp=*head;
      while(temp->next!=NULL)
      {
```

Scanned with OKEN Scanner

```
        temp=temp->next;
      }
      temp->next=newnode;
    }
  }
  void printlist(node* head)
  {
    node*temp=head;
    while(temp!=NULL){
      printf("%d",temp->data);
      temp=temp->next;
    }
  }
  int main(){
    int n,m,v;
    node*mor=NULL;
    node*eve=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
      scanf("%d",&v);
      insert(&mor,v);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++){
      scanf("%d",&v);
      insert(&eve,v);
    }
    if(mor==eve){
      mor=eve;
    }
    else
    {
      node*temp=mor;
      while(temp->next!=NULL)
      {
        temp=temp->next;
      }
      temp->next=eve;

    }
    printlist(mor);
    return 0;
```

240701244

Scanned with OKEN Scanner

}

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

3. 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".

Scanned with OKEN Scanner

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;
void display(){
    if(head==NULL){
```

240701244

Scanned with OKEN Scanner

```
        printf("The list is empty\n");
        return;
    }
    node*temp=head;
    while(temp){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
void createlist(){

    int value;
    node*temp,*newnode;
    head=NULL;
    while(1){
        scanf("%d",&value);
        if(value==-1)
        break;
    newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    newnode->next=NULL;
    if(head==NULL)
    {
        head=newnode;
        temp=head;
    }
    else
    {
        temp->next=newnode;
        temp=newnode;
    }
    }
    printf("LINKED LIST CREATED\n");
}
void insertbeg(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:\n");
    display();
```

240701244

Scanned with OKEN Scanner

```
}
void insertend(int value)
{
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    newnode->next=NULL;
    if(head==NULL){
        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:\n");
    display();
}
void insertbefore(int tar,int value){
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    if(head->data==tar){
        newnode->next=head;
        head=newnode;
        printf("The linked list after insertion before a value is:\n");
        display();
        return;
    }
    node*temp=head;
    while(temp->next&&temp->next->data!=tar)
    {
        temp=temp->next;
    }
    if(temp->next==NULL){
        printf("Value not found in the list\n");
        printf("The linked list after insertion before a value is:\n");
        display();
        return;
    }
```

240701244

Scanned with OKEN Scanner

```
    newnode->next=temp->next;
    temp->next=newnode;
    printf("The linked list after insertion before a value is:\n");
    display();
}
void insertafter(int tar,int value){
    node*temp=head;
    while(temp&&temp->data!=tar){
        temp=temp->next;
    }
    if(temp==NULL){
        printf("Value not found in the list\n");
        printf("The linked list after insertion after a value is:\n");
        display();
        return;
    }
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    newnode->next=temp->next;
    temp->next=newnode;
    printf("The linked list after insertion after a value is:\n");
    display();
}
void deletebeg(){
    if(head==NULL){
        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:\n");
    display();

}
void deleteend(){
    if(head==NULL){
        printf("The list is empty\n");
        return;
    }
    node*temp=head;
    if(head->next==NULL){
```

240701244

Scanned with OKEN Scanner

```
        free(head);
        head=NULL;
    }
    else
    {
        while(temp->next->next){
            temp=temp->next;
        }
        free(temp->next);
        temp->next=NULL;
    }
    printf("The linked list after deletion from the end is:\n");
    display();

}
void deletebefore(int tar)
{
    if(head==NULL||head->next==NULL||head->data==tar){
        printf("Value not found in list\n");
        return;
    }
    node*prev=NULL,*temp=head,*beforetemp=NULL;
    if(head->next->data==tar){
        temp=head;
        head=head->next;
        free(temp);
        printf("The linked list after deletion before a value is:\n");
        display();
        return;
    }
    while(temp->next&&temp->next->data!=tar)
    {
        beforetemp=prev;
        prev=temp;
        temp=temp->next;
    }
    if(temp->next==NULL){
        printf("Value not found in the list\n");
        return;
    }
    if(beforetemp==NULL){
        head=temp;
```

240701244

Scanned with OKEN Scanner

```
    }
    else{
        beforetemp->next=temp;
    }
    free(prev);
    printf("The linked list after deletion before a value is:\n");
    display();
}
void deleteafter(int tar)
{
    node*temp=head;
    while(temp&&temp->data!=tar){
        temp=temp->next;
    }
    if(temp==NULL||temp->next==NULL)
    {
        printf("Value not found in the list\n");
        return;
    }
    node*deletenode=temp->next;
    temp->next=temp->next->next;
    free(deletenode);
    printf("The linked list after deletion after a value is:\n");
    display();
}
int main(){
    int c,v,t;
    while(1)
    {
        scanf("%d",&c);
        switch(c){
            case 1:
            createlist();
            break;
            case 2:
            display();
            break;
            case 3:
            scanf("%d",&v);
            insertbeg(v);
            break;
```

240701244

Scanned with OKEN Scanner

```
        case 4:
        scanf("%d",&v);
        insertend(v);
        break;
        case 5:
        scanf("%d %d",&t,&v);
        insertbefore(t,v);
        break;
        case 6:
        scanf("%d %d",&t,&v);
        insertafter(t,v);
        break;
        case 7:
        deletebeg();
        break;
        case 8:
        deleteend();
        break;
        case 9:
        scanf("%d",&t);
        deletebefore(t);
        break;
        case 10:
        scanf("%d",&t);
        deleteafter(t);
        break;
        case 11:
        exit(0);
        default:
        printf("Invalid option! Please try again\n");
    }
    }
    return 0;
}
```

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

4. 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

Scanned with OKEN Scanner

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.

Scanned with OKEN Scanner

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;
void display(){
    if(head==NULL){
        printf("The list is empty\n");
        return;
    }
    node*temp=head;
    while(temp){
        printf("%d ",temp->data);
        temp=temp->next;
```

Scanned with OKEN Scanner

```
    }
    printf("\n");
}
void createlist(){
    int value;
    node*temp,*newnode;
    head=NULL;
    while(1){
        scanf("%d",&value);
        if(value==-1)
        break;
    newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    newnode->next=NULL;
    if(head==NULL){
        head=newnode;
        temp=head;
    }
    else
    {
        temp->next=newnode;
        temp=newnode;
    }
    }
    printf("LINKED LIST CREATED\n");

}
void insertbeg(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:\n");
    display();
}
void insertend(int value){
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    newnode->next=NULL;
    if(head==NULL){
        head=newnode;
    }
```

240701244

Scanned with OKEN Scanner

```
    else
    {
        node*temp=head;
        while(temp->next){
            temp=temp->next;
        }
        temp->next=newnode;
    }
    printf("The linked list after insertion at the end is:\n");
    display();
}
void insertbefore(int tar,int value){
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=value;
    if(head->data==tar)
    {
        newnode->next=head;
        head=newnode;
        printf("The linked list after insertion before a value is:\n");
        display();
        return;
    }
    node*temp=head;
    while(temp->next&&temp->next->data!=tar){
        temp=temp->next;
    }
    if(temp->next==NULL){
        printf("Value not found in the list\n");
        printf("The linked list after insertion before a value is:\n");
        display();
        return;
    }
    newnode->next=temp->next;
    temp->next=newnode;
    printf("The linked list after insertion before a value is:\n");
    display();

}
void insertafter(int tar,int value){
    node*temp=head;
    while(temp&&temp->data!=tar){
        temp=temp->next;
```

240701244

Scanned with OKEN Scanner

```
}
if(temp==NULL){
    printf("Value not found in the list\n");
    printf("The linked list after insertion after a value is:\n");
    display();
    return;
}
node*newnode=(node*)malloc(sizeof(node));
newnode->data=value;
newnode->next=temp->next;
temp->next=newnode;
printf("The linked list after insertion after a value is:\n");
display();
}
void deletebeg(){
    if(head==NULL){
        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:\n");
    display();
}
void deleteend()
{
    if(head==NULL){
        printf("The list is empty\n");
        return;
    }
    node*temp=head;
    if(head->next==NULL){
        free(head);
        head=NULL;
    }
    else
    {
        while(temp->next->next)
        {
            temp=temp->next;
        }
```

240701244

Scanned with OKEN Scanner

```
        free(temp->next);
        temp->next=NULL;
    }
    printf("The linked list after deletion from the end is:\n");
    display();
}
void deletebefore(int tar){
    if(head==NULL||head->next==NULL||head->data==tar)
    {
        printf("The list is empty\n");
        return;
    }
    node*prev=NULL,*temp=head,*beforetemp=NULL;
    if(head->next->data==tar){
        temp=head;
        head=head->next;
        free(temp);
        printf("The linked list after deletion before a value is:\n");
        display();
        return;
    }
    while(temp->next&&temp->next->data!=tar){
        beforetemp=prev;
        prev=temp;
        temp=temp->next;

    }
    if(temp->next==NULL)
    {
        printf("Value not found in the list\n");
        return;
    }
    if(beforetemp==NULL){
        head=temp;
    }
    else
    {
        beforetemp->next=temp;
    }
    free(prev);
    printf("The linked list after deletion before a value is:\n");
    display();
```

240701244

Scanned with OKEN Scanner

```
}
void deleteafter(int tar){
    node*temp=head;
    while(temp&&temp->data!=tar){
        temp=temp->next;
    }
    if(temp==NULL||temp->next==NULL){
        printf("Value not found in the list\n");
        return;
    }
    node*deletenode=temp->next;
    temp->next=temp->next->next;
    free(deletenode);
    printf("The linked list after deletion after a value is:\n");
    display();
}
int main(){
    int c,v,t;
    while(1){
        scanf("%d",&c);
        switch(c){
            case 1:
            createlist();
            break;
            case 2:
            display();
            break;
            case 3:
            scanf("%d",&v);
            insertbeg(v);
            break;
            case 4:
            scanf("%d",&v);
            insertend(v);
            break;
            case 5:
            scanf("%d %d",&t,&v);
            insertbefore(t,v);
            break;
            case 6:
            scanf("%d %d",&t,&v);
            insertafter(t,v);
```

240701244

Scanned with OKEN Scanner

```
            break;
            case 7:
            deletebeg();
            break;
            case 8:
            deleteend();
            break;
            case 9:
            scanf("%d",&t);
            deletebefore(t);
            break;
            case 10:
            scanf("%d",&t);
            deleteafter(t);
            break;
            case 11:
            exit(0);
            break;
            default:
            printf("Invalid option! Please try again\n");
        }
    }
    return 0;
}
```

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

5. 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

Scanned with OKEN Scanner

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.

240701244

Scanned with OKEN Scanner

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 co,ex;
    struct Node*next;
}node;
node*create(int c,int e){
    node*newnode=(node*)malloc(sizeof(node));
    newnode->co=c;
    newnode->ex=e;
    newnode->next=NULL;
    return newnode;
}
void insert(node** head,int c,int e){
    node*newnode=create(c,e);
    if(*head==NULL)
    {
        *head=newnode;
    }
    else
    {
        node*temp=*head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
    }
}
```

Scanned with OKEN Scanner

```
int evaluate(node*head,int x){
    node*temp=head;
    int result=0;
    while(temp!=NULL){
        result+=temp->co*(int)pow(x,temp->ex);
        temp=temp->next;
    }
    return result;
}
int main()
{
    node*poly=NULL;
    int n,co,x;
    scanf("%d",&n);
    for(int i=n;i>=0;i--){
        scanf("%d",&co);
        insert(&poly,co,i);
    }
    scanf("%d",&x);
    int result=evaluate(poly,x);
    printf("%d",result);
    return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 2
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

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

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

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = NULL;
```

Scanned with OKEN Scanner

```
    temp->prev = NULL;
    head = temp;
    printf("%d\n", head->data);
    free(temp);
    return 0;
}
```

Answer

2

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

2. Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

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

3. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

void addFirst(int data){ Node* newNode = new Node(data); newNode->next = head; if (head != NULL) { head->prev = newNode; } head = newNode; }

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

4. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

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

5. Consider the provided pseudo code. How can you initialize an empty

Scanned with OKEN Scanner

two-way linked list?

```
Define Structure Node
   data: Integer
   prev: Pointer to Node
   next: Pointer to Node
End Define

Define Structure TwoWayLinkedList
   head: Pointer to Node
   tail: Pointer to Node
End Define
```

Answer

struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list->head = NULL; list->tail = NULL;

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

6. Consider the following function that refers to the head of a Doubly Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

```
Procedure fun(head_ref: Pointer to Pointer of node)
   temp = NULL
   current = *head_ref

   While current is not NULL
      temp = current->prev
      current->prev = current->next
      current->next = temp
      current = current->prev
   End While

   If temp is not NULL
```

Scanned with OKEN Scanner

```
    *head_ref = temp->prev
  End If
End Procedure
```

Answer

6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.

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

7. How many pointers does a node in a doubly linked list have?

Answer

2

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

8. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

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

9. What is a memory-efficient double-linked list?

Answer

A doubly linked list that uses bitwise AND operator for storing addresses

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

10. What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

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

Scanned with OKEN Scanner

11. Which of the following information is stored in a doubly-linked list's nodes?

Answer

All of the mentioned options

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

12. Which of the following is false about a doubly linked list?

Answer

Implementing a doubly linked list is easier than singly linked list

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

13. Which of the following is true about the last node in a doubly linked list?

Answer

Its next pointer is NULL

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

14. Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {
    if (*head_ref == NULL || del_node == NULL) {
        return;
    }
    if (*head_ref == del_node) {
        *head_ref = del_node->next;
    }
    if (del_node->next != NULL) {
        del_node->next->prev = del_node->prev;
    }
    if (del_node->prev != NULL) {
```

240701244

Scanned with OKEN Scanner

```
    del_node->prev->next = del_node->next;
  }
  free(del_node);
}
```

Answer

Deletes the first occurrence of a given data value in a doubly linked list.

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

15. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

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

16. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));

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

17. What will be the output of the following program?

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

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

int main() {
```

Scanned with OKEN Scanner

```
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < 5; i++) {
        struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
        temp->data = i + 1;
        temp->prev = tail;
        temp->next = NULL;
        if (tail != NULL) {
            tail->next = temp;
        } else {
            head = temp;
        }
        tail = temp;
    }
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    return 0;
}
```

Answer

1 2 3 4 5

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

18. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {
    int Value;
    struct Node *Fwd;
    struct Node *Bwd;
};
```

Scanned with OKEN Scanner

Answer

X->Bwd->Fwd = X->Fwd; X->Fwd->Bwd = X->Bwd;

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

19. What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
newNode->prev = NULL;
```

Answer

Creates a new node and initializes its data to 'value'

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

20. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

Here are the main functionalities of the program:

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

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

Input Format

Scanned with OKEN Scanner

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

The input is terminated by entering - (hyphen).

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

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

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

// You are using GCC
typedef struct Node Node;
Node *create(char x)
{
   Node *newnode;
   newnode=(Node *)malloc(sizeof(Node));
   newnode->item=x;
   newnode->next=NULL;
   newnode->prev=NULL;
   return newnode;
```

Scanned with OKEN Scanner

```
}
void insertAtEnd(struct Node** head, char item) {
    Node *newnode=create(item);
    if(*head==NULL){
        *head=newnode;
        return;
    }
    Node *current=*head;
    while(current->next!=NULL){
        current=current->next;
    }
    newnode->prev=current;
    current->next=newnode;
    return;
}
void displayForward(struct Node* head) {
    //type your code here
    Node *current=head;
    while(current!=NULL){
        printf("%c ",current->item);
        current=current->next;
    }
    printf("\n");
    return;
}

void displayBackward(struct Node* tail) {
    //type your code here
    Node *current=tail;
    while(tail!=NULL){
        printf("%c ",tail->item);
        tail=tail->prev;
    }
    printf("\n");
    return;
}

void freePlaylist(struct Node* head) {
    //type your code here
    while(head!=NULL){
        Node *temp=head;
        head=head->next;
```

240701244

Scanned with OKEN Scanner

```
        free(temp);
    }
    return;
}

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

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

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

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

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

    freePlaylist(playlist);

    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Scanned with OKEN Scanner

Output Format

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155
Output: 163

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
typedef struct node Node;
Node *create(int x){
   Node *newnode;
   newnode=(Node *)malloc(sizeof(Node));
   newnode->data=x;
   newnode->prev=NULL;
   newnode->next=NULL;
   return newnode;
}
void insert_end(Node **head,int x){
   Node *newnode=create(x);
   if(*head==NULL){
      *head=newnode;
      return;
   }
   Node *current=*head;
```

240701244

Scanned with OKEN Scanner

```
    while(current->next!=NULL){
        current=current->next;
    }
    newnode->prev=current;
    current->next=newnode;
}
void find_large(Node **head){
    if(*head==NULL || (*head)->data==0){
        printf("Empty list!");
        return;
    }
    Node *current=*head;
    Node *max=current;
    while(current!=NULL){
        if(current->data > max->data){
            max=current;
        }
        current=current->next;
    }
    printf("%d",max->data);
    return;
}
int main(){
    Node *head=NULL;
    int n,x;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        insert_end(&head,x);
    }
    find_large(&head);
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bob is tasked with developing a company's employee record management system. The system needs to maintain a list of employee records using a doubly linked list. Each employee is represented by a unique integer ID.

Help Bob to complete a program that adds employee records at the front, traverses the list, and prints the same for each addition of employees to the list.

Input Format

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

The second line consists of N space-separated integers, representing the employee IDs.

Scanned with OKEN Scanner

Output Format

For each employee ID, the program prints "Node Inserted" followed by the current state of the doubly linked list in the next line, with the data values of each node separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
101 102 103 104
Output: Node Inserted
101
Node Inserted
102 101
Node Inserted
103 102 101
Node Inserted
104 103 102 101

Answer

```
#include <iostream>
using namespace std;

struct node {
   int info;
   struct node* prev, * next;
};

struct node* start = NULL;

// You are using GCC
typedef struct node Node;
void traverse() {
   //type your code here
   Node *current=start;
   printf("Node Inserted\n");
   while(current!=NULL){
      printf("%d ",current->info);
      current=current->next;
```

Scanned with OKEN Scanner

```
    }
    printf("\n");
    return;
}

void insertAtFront(int data) {
    //type your code here
    Node *newnode;
    newnode=(Node *)malloc(sizeof(Node));
    newnode->info=data;
    if(start==NULL){
        start=newnode;
        return;
    }
    newnode->prev=NULL;
    start->prev=newnode;
    newnode->next=start;
    start=newnode;
}

int main() {
    int n, data;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> data;
        insertAtFront(data);
        traverse();
    }
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

Input Format

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

Scanned with OKEN Scanner

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
typedef struct node Node;
Node *create(int x){
   Node *newnode;
   newnode=(Node *)malloc(sizeof(Node));
   newnode->data=x;
   newnode->prev=NULL;
   newnode->next=NULL;
   return newnode;
}
void insert_end(Node **head,int x)
{
   Node *newnode;
   newnode=create(x);
   if(*head==NULL){
      *head=newnode;
```

Scanned with OKEN Scanner

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

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ashwin is tasked with developing a simple application to manage a list of items in a shop inventory using a doubly linked list. Each item in the inventory has a unique identification number. The application should allow users to perform the following operations:

Create a List of Items: Initialize the inventory with a given number of items. Each item will be assigned a unique number provided by the user and insert the elements at end of the list.

Delete an Item: Remove an item from the inventory at a specific position.

Display the Inventory: Show the list of items before and after deletion.

If the position provided for deletion is invalid (e.g., out of range), it should

Scanned with OKEN Scanner

display an error message.

Input Format

The first line contains an integer n, representing the number of items to be initially entered into the inventory.

The second line contains n integers, each representing the unique identification number of an item separated by spaces.

The third line contains an integer p, representing the position of the item to be deleted from the inventory.

Output Format

The first line of output prints "Data entered in the list:" followed by the data values of each node in the doubly linked list before deletion.

If p is an invalid position, the output prints "Invalid position. Try again."

If p is a valid position, the output prints "After deletion the new list:" followed by the data values of each node in the doubly linked list after deletion.

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: 4
1 2 3 4
5
Output: Data entered in the list:
 node 1 : 1
 node 2 : 2
 node 3 : 3
 node 4 : 4
Invalid position. Try again.
```

Answer

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

240701244

Scanned with OKEN Scanner

```
typedef struct Node{
   int data;
   struct Node* prev;
   struct Node* next;
}Node;
void append(Node**head,int value){
   Node*newnode=(Node*)malloc(sizeof(Node));
   newnode->data=value;
   newnode->next=NULL;
   if(*head==NULL){
      newnode->prev=NULL;
      *head=newnode;
      return;
   }
   Node*temp=*head;
   while(temp->next!=NULL)
     temp=temp->next;
    temp->next=newnode;
    newnode->prev=temp;
}
void display(Node*head){
   Node*temp=head;
   int nodeNum=1;
   while(temp!=NULL){
      printf(" node %d : %d\n",nodeNum,temp->data);
      temp=temp->next;
      nodeNum++;
   }
}
int deleteAtPosition(Node**head,int pos,int totalNodes){
   if(pos<1||pos>totalNodes)
      return 0;
   Node*temp=*head;
   for(int i=1;i<pos;i++)
      temp=temp->next;
   if(temp->prev!=NULL)
      temp->prev->next=temp->next;
   else
     *head=temp->next;
   if(temp->next!=NULL)
         temp->next->prev=temp->prev;
   free(temp);
```

Scanned with OKEN Scanner

```
    return 1;
}
int main()
{
    int n,position,value;
    Node*head=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&value);
        append(&head,value);
    }
    scanf("%d",&position);
    printf("Data entered in the list:\n");
    display(head);
    if(position <1||position >n){
        printf("Invalid position. Try again.");

    }
    else{
        deleteAtPosition(&head,position,n);
        printf("\nAfter deletion the new list:\n");
        display(head);
    }
    return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

Input Format

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

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

Scanned with OKEN Scanner

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8
Output: 8 4 6 10 12

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
   struct Node*prev;
   int data;
   struct Node*next;
}node;
node*head=NULL;
void insert(int e)
{
   node*newnode=(node*)malloc(sizeof(node));
   newnode->data=e;
   newnode->prev=NULL;
   newnode->next=head;
   if(head!=NULL){
      head->prev=newnode;
   }
   head=newnode;

}
void removeduplicates()
{
   node*current=head;
   while(current!=NULL)
   {
```

Scanned with OKEN Scanner

```
        node*runner=current->next;
        while(runner!=NULL)
        {
            node*nextnode=runner->next;
            if(runner->data==current->data)
            {
                if(runner->next!=NULL)
                {
                    runner->next->prev=runner->prev;
                }
                if(runner->prev!=NULL)
                {
                    runner->prev->next=runner->next;
                }
                free(runner);

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

240701244

Scanned with OKEN Scanner

}

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>
```

Scanned with OKEN Scanner

```
#include<stdlib.h>
typedef struct Node
{
   struct Node*prev;
   int data;
   struct Node *next;
}node;
node*head=NULL;
void insert(int e)
{
   node*newnode=(node*)malloc(sizeof(node));
   newnode->data=e;
   newnode->prev=NULL;
   newnode->next=NULL;
   if(head==NULL)
   {
      head=newnode;
   }
   else
   {
      node*temp=head;
      while(temp->next!=NULL)
      {
         temp=temp->next;
      }
      temp->next=newnode;
      newnode->prev=temp;
   }
}
void reverse()
{
   node*temp=NULL;
   node*current=head;
   while(current!=NULL)
   {
      temp=current->prev;
      current->prev=current->next;
      current->next=temp;
      current=current->prev;
   }
   if(temp!=NULL){
      head=temp->prev;
```

240701244

Scanned with OKEN Scanner

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

Status : Correct

Marks : 10/10

3. Problem Statement

Imagine Anu is tasked with finding the middle element of a doubly linked list. Given a doubly linked list where each node contains an integer value and is inserted at the end, implement a program to find the middle element of the list. If the number of nodes is even, return the middle element pair.

Input Format

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

Scanned with OKEN Scanner

The second line consists of N space-separated integers, representing the values of the nodes in the doubly linked list.

Output Format

The first line of output prints the space-separated elements of the doubly linked list.

The second line prints the middle element(s) of the doubly linked list, depending on whether the number of nodes is odd or even.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50
30

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
    struct Node*prev;
    int data;
    struct Node*next;
}node;
node*head=NULL;
void insert(int e)
{
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=e;
    newnode->next=NULL;
    newnode->prev=NULL;
    if(head==NULL)
    {
        head=newnode;
```

Scanned with OKEN Scanner

```
    }
    else
    {
        node*temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        newnode->prev=temp;
    }
}
void middle()
{
    node*fast=head;
    node*slow=head;
    node*prev=NULL;
    while(fast!=NULL&&fast->next!=NULL)
    {
        fast=fast->next->next;
        prev=slow;
        slow=slow->next;
    }
    if(fast==NULL){
        printf("%d %d",prev->data,slow->data);
    }
    else
    {
        printf("%d",slow->data);
    }

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

240701244

Scanned with OKEN Scanner

```
{
    int n,e;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        insert(e);
    }
    display();
    middle();
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

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.

Scanned with OKEN Scanner

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>
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
typedef struct node Node;
Node*create(int x)
{
    Node*newnode;
    newnode=(Node*)malloc(sizeof(Node));
    newnode->data=x;
    newnode->next=NULL;
    newnode->prev=NULL;
    return newnode;
}
```

Scanned with OKEN Scanner

```
void insertend(Node**head,int x)
{
    Node *newnode=create(x);
    if(*head==NULL){
        *head=newnode;
        return;
    }
    newnode->next=(*head);
    (*head)->prev=newnode;
    *head=newnode;
    return;
}
void deleteposition(Node**head,int pos){
    Node *current=*head;
    if(pos==1){
        Node*temp=*head;
        (*head)->prev=NULL;
        (*head)=(*head)->next;
        free(temp);
        return;
    }
    int count=1;
    while(current->next!=NULL && count<pos){
        current=current->next;
        count++;
    }
    if(current->next==NULL)
    {
        Node*temp1=current;
        temp1->prev->next=NULL;
        free(temp1);
        return;
    }
    Node*temp=current;
    current=current->next;
    current->prev=temp->prev;
    temp->prev->next=current;
    free(temp);
    return;
}
void display(Node *head){
    Node *current=head;
```

Scanned with OKEN Scanner

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

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

2. 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.

Scanned with OKEN Scanner

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>
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
typedef struct node Node;
Node *create(int x)
{
   Node *newnode;
   newnode=(Node*)malloc(sizeof(Node));
   newnode->prev=NULL;
   newnode->next=NULL;
   newnode->data=x;
   return newnode;
}
void insertend(Node **head,int x){
   Node *newnode=create(x);
   if(*head==NULL){
      *head=newnode;
      return;
   }
```

Scanned with OKEN Scanner

```
    else
    {
        Node *current=*head;
        while(current->next!=NULL){
            current=current->next;
        }
        newnode->prev=current;
        current->next=newnode;
        return;
    }

}
void rotate(Node**head,int k){
    if(*head==NULL || k==0){
        return;
    }
    int count=1;
    Node*tail=*head;
    while(tail->next!=NULL){
        tail=tail->next;
        count++;
    }
    Node*newtail=*head;
    int rp=count-k;
    for(int i=1;i<rp;i++){
        newtail=newtail->next;
    }
    Node*newhead=newtail->next;
    newhead->prev=NULL;
    newtail->next=NULL;
    tail->next=*head;
    (*head)->prev=tail;
    *head=newhead;
    return;
}
void display(Node*head)
{
    Node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;
    }
```

240701244

Scanned with OKEN Scanner

```
    printf("\n");
    return;
}
int main()
{
    Node*head=NULL;
    int n,x,k;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        insertend(&head,x);
    }
    scanf("%d",&k);
    rotate(&head,k);
    display(head);
}
```

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

3. 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).

Scanned with OKEN Scanner

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>
struct node
{
    int data;
    struct node *prev;
    struct node *next;
};
typedef struct node Node;
void insert_at_end(Node **head,int x)
{
    Node *newnode;
    newnode=(Node *)malloc(sizeof(Node));
    newnode->data=x;
    newnode->next=NULL;
    if((*head)==NULL){
```

Scanned with OKEN Scanner

```
        newnode->next=*head;
        *head=newnode;
        return;
    }
    newnode->next=*head;
    (*head)->prev=newnode;
    *head=newnode;
    return;
}
void insert_position(Node **head,int pos,int x){
    Node *newnode;
    newnode=(Node *)malloc(sizeof(Node));
    newnode->data=x;
    if(pos==1){
        newnode->next=*head;
        (*head)->prev=newnode;
        *head=newnode;
        return;
    }
    Node *temp=*head;
    int count=1;
    while(temp!=NULL&&count<pos-1){
        temp=temp->next;
        count++;
    }
    newnode->prev=temp;
    newnode->next=temp->next;
    temp->next=newnode;
    return;
}
void display(Node*head){
    Node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;
    }
    printf("\n");
    return;
}
int main()
{
    Node *head=NULL;
```

240701244

Scanned with OKEN Scanner

```
    int n,x,pos;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        insert_at_end(&head,x);
    }
    display(head);
    int value;
    scanf("%d",&pos);
    scanf("%d",&value);
    insert_position(&head,pos,value);
    display(head);
}
```

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

4. 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.

Scanned with OKEN Scanner

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>
struct node
{
   int data;
   struct node *prev;
   struct node*next;
};
typedef struct node Node;
Node *create(int x){
   Node *newnode;
   newnode=(Node*)malloc(sizeof(Node));
   newnode->data=x;
   newnode->prev=NULL;
   newnode->next=NULL;
   return newnode;
}
void insertend(Node **head,int x){
   Node *newnode=create(x);
   if(*head==NULL){
      *head=newnode;
      return;
```

240701244

Scanned with OKEN Scanner

```
    }
    else
    {
      Node *current=*head;
      while(current->next!=NULL)
      {
         current=current->next;
      }
      current->next=newnode;
      newnode->prev=current;
    }
    return;
}
void findmid(Node*head){
    Node *fast=head;
    Node *slow=head;
    while(fast!=NULL && fast->next!=NULL){
        fast=fast->next->next;
        if(fast!=NULL){
            slow=slow->next;
        }
    }
    if(fast==NULL){
        printf("%d %d",slow->data,slow->next->data);
    }
    else
    {
        printf("%d",slow->data);
    }
    printf("\n");
    return;
}
void display(Node *head){
    Node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;

    }
    printf("\n");
    return;
}
```

240701244

Scanned with OKEN Scanner

```
int main()
{
    Node *head=NULL;
    int n,x;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        insertend(&head,x);
    }
    display(head);
    findmid(head);
}
```

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".

Scanned with OKEN Scanner

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>
struct node
{
    int data;
    struct node *prev;
    struct node *next;
};
typedef struct node Node;
Node*create(int x){
    Node *newnode;
    newnode=(Node*)malloc(sizeof(Node));
    newnode->data=x;
    newnode->prev=NULL;
    newnode->next=NULL;
    return newnode;
}
void insertend(Node **head,int x){
    Node *newnode=create(x);
    if(*head==NULL){
        *head=newnode;
        return;
    }
    else{
        Node*current=*head;
        while(current->next!=NULL){
            current=current->next;
        }
        current->next=newnode;
```

Scanned with OKEN Scanner

```
        newnode->prev=current;
        return;
    }
}
int palindrome(Node *head){
    Node *start=head;
    Node *current=head;
    while(current->next!=NULL){
        current=current->next;
    }
    while(start!=current && start->prev!=current){
        if(start->data!=current->data){
            return 0;
        }
        start=start->next;
        current=current->prev;
    }
    return 1;
}
void display(Node *head){
    Node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;
    }
    printf("\n");
    return;
}
int main(){
    int n,x;
    Node *head=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&x);
        insertend(&head,x);
    }
    display(head);
    if(palindrome(head)){
        printf("The doubly linked list is a palindrome");
    }
    else{
        printf("The doubly linked list is not a palindrome");
```

240701244

Scanned with OKEN Scanner

```
    }
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

1. 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***

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

Answer

142

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

Scanned with OKEN Scanner

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

Answer

-18

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

4. 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***

5. Elements are Added on _____ of the Stack.

Answer

Top

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

6. 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;
```

Scanned with OKEN Scanner

```
    }
    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***

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

Answer

All of the mentioned options

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

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

Answer

Efficient memory usage

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

9. A user performs the following operations on stack of size 5 then which

Scanned with OKEN Scanner

of the following is correct statement for Stack?

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

Answer

Underflow Occurs

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

10. 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() {
```

Scanned with OKEN Scanner

```
    printf("%d\n", isEmpty());
    push(10);
    push(20);
    push(30);
    printf("%d\n", isFull());
    return 0;
}
```

Answer

10

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

11. 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***

12. 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***

13. 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***

Scanned with OKEN Scanner

14. 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***

15. 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***

16. 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 {
```

Scanned with OKEN Scanner

```
        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***

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

Answer

Pop

Scanned with OKEN Scanner

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

18. 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***

19. 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***

20. 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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Scanned with OKEN Scanner

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".

240701244

Scanned with OKEN Scanner

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;

// You are using GCC
typedef struct Node Node;
Node* list=NULL;
void push(int value) {
   //Type your code here
   Node* newnode;
   newnode=(Node *)malloc(sizeof(Node));
   newnode->data=value;
   if(list==NULL){
```

Scanned with OKEN Scanner

```
        newnode->next=NULL;
    }
    else
    {
        newnode->next=list;
    }
    list=newnode;
    printf("Pushed element: %d",newnode->data);
    printf("\n");
}

void pop() {
    //Type your code here
    if(list!=NULL){
        Node *temp=list;
        printf("Popped element: %d",temp->data);
        list=list->next;
        free(temp);
    }
    else{
        printf("Stack is empty. Cannot pop.\n");
    }
    printf("\n");
    return;
}

void displayStack() {
    //Type your code here
    if(list!=NULL){
        Node*current=list;
        printf("Stack elements (top to bottom): ");
        while(current!=NULL){
            printf("%d ",current->data);
            current=current->next;
        }
        printf("\n");
    }
    else{
        printf("Stack is empty\n");
    }
}
int main() {
```

240701244

Scanned with OKEN Scanner

```
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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 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

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

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>
#include<stdlib.h>
struct node
{
   int data;
   struct node*next;
}*list=NULL;
typedef struct node Node;
void push(int x)
{
   Node *newnode;
   newnode=(Node*)malloc(sizeof(Node));
   newnode->data=x;
   printf("BOOK ID %d is pushed onto the stack\n",newnode->data);
   if(list!=NULL)
   {
      newnode->next=list;
      list=newnode;
      return;
   }
   newnode->next=NULL;
   list=newnode;
   return;
}
void pop()
{
   if(list!=NULL){
      Node* temp=list;
      list=list->next;
      printf("Book ID %d is popped from the stack",temp->data);
   }
   else{
      printf("Stack Underflow");
   }
   printf("\n");
```

Scanned with OKEN Scanner

```
    return;
}
void display(){
    if(list!=NULL){
        printf("Book ID in the stack: ");
        Node *pos=list;
        while(pos!=NULL){
            printf("%d ",pos->data);
            pos=pos->next;
        }
        printf("\n");
    }
    else{
        printf("Stack is empty\n");
    }
    return;
}
int main()
{
    int choice,x;
    while(1){
        scanf("%d",&choice);
        switch(choice){
            case 1:
            scanf("%d",&x);
            push(x);
            break;
            case 2:
            pop();
            break;
            case 3:
            display();
            break;
            case 4:
            printf("Exiting the program\n");
            return 0;
            default:
            printf("Invalid choice\n");
            break;
        }
    }
}
```

240701244

Scanned with OKEN Scanner

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

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

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

Input Format

Scanned with OKEN Scanner

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>
```

240701244

Scanned with OKEN Scanner

```c
#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;
}

// You are using GCC
void push(char value) {
   //Type your code here
   top=top+1;
   items[top]=value;
   printf("Pushed: %c\n",value);
}

char pop() {
   //Type your code here
   if(top==-1){
      printf("Stack is empty. Nothing to pop.\n");
      return '\0';
   }
   else{
      char ans=items[top];
      printf("Popped: %c\n",ans);
      top=top-1;
      return ans;
   }
}
void display() {
   //Type your code here
   if(top!=-1){
      int a=top;
```

240701244

Scanned with OKEN Scanner

```
        printf("Stack elements: ");
        while(a!=-1)
        {
            printf("%c ",items[a]);
            a=a-1;
        }
        printf("\n");
    }
    else{
        printf("Stack is empty.\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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Example

Scanned with OKEN Scanner

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)
```

240701244

Scanned with OKEN Scanner

```
        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
int isOperand(char ch) {
    //type your code here
    return(ch>='a'&&ch<='z')||(ch>='A' && ch<='Z');
}

int Prec(char ch) {
    //type your code here
    if(ch=='^')
        return 3;
    if(ch=='*'||ch=='/')
        return 2;
    if(ch=='+'||ch=='-')
        return 1;
    return -1;
}
```

240701244

Scanned with OKEN Scanner

```
void infixToPostfix(char* exp) {
    struct Stack*stack=createStack(strlen(exp));
    if(!stack)
        return;
    char result[strlen(exp)+1];
    int k=0;
    for(int i=0;exp[i];i++){
        char current=exp[i];
        if(isOperand(current)){
            result[k++]=current;
        }
        else if(current=='('){
            push(stack,current);
        }
        else if(current==')'){
            while(!isEmpty(stack)&&peek(stack)!='('){
                result[k++]=pop(stack);
            }
            pop(stack);
        }
        else{
            while(!isEmpty(stack)&&Prec(peek(stack))>=Prec(current)){
                result[k++]=pop(stack);
            }
            push(stack,current);
        }
    }
    while(!isEmpty(stack)){
        result[k++]=pop(stack);
    }
    result[k]='\0';
    printf("%s\n",result);
}

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

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

240701244

Scanned with OKEN Scanner

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

Scanned with OKEN Scanner

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

Choice 2: Pop class schedule from the stack

Choice 3: Display the class schedules in the stack.

Choice 4: Exit the program.

Output Format

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

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

Refer to the sample output for the exact format.

Sample Test Case

Input: 1 d
1 h
3
2

Scanned with OKEN Scanner

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

Answer

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

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

struct Node* top = NULL;

// You are using GCC
void push(char value) {
    //Type your code here
    struct Node*newNode=(struct Node*)malloc(sizeof(struct Node));
    if(!newNode){
        printf("Memory allocation failed\n");
        return;
    }
    newNode->data=value;
    newNode->next=top;
    top=newNode;
    printf("Adding Section: %c\n",value);
}

void pop() {
    //Type your code here
    if(top==NULL){
        printf("Stack is empty. Cannot pop.\n");
    }else{
        struct Node*temp=top;
        printf("Removing Section: %c\n",temp->data);
        top=top->next;
```

Scanned with OKEN Scanner

```
        free(temp);
    }
}

void displayStack() {
    //Type your code here
    if(top==NULL){
        printf("Stack is empty\n");
    }else{
        printf("Enrolled Sections: ");
        struct Node*current=top;
        while(current!=NULL){
            printf("%c ",current->data);
            current=current->next;
        }
        printf("\n");
    }
}

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

240701244

Scanned with OKEN Scanner

```
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

In an educational setting, Professor Smith tasks Computer Science students with designing an algorithm to evaluate postfix expressions efficiently, fostering problem-solving skills and understanding of stack-based computations.

The program prompts users to input a postfix expression, evaluates it, and displays the result, aiding students in honing their coding abilities.

Input Format

The input consists of the postfix mathematical expression.

The expression will contain real numbers and mathematical operators (+, -, *, /), without any space.

Scanned with OKEN Scanner

Output Format

The output prints the result of evaluating the given postfix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 82/
Output: 4

Answer

```
// You are using GCC
#include<stdio.h>
#include<ctype.h>
#define MAX 100
float stack[MAX];
int top=-1;
void push(float val){
    if(top==MAX-1){
        return;
    }
    stack[++top]=val;
}
float pop(){
    if(top==-1){
        return -1;
    }
    return stack[top--];
}
int main(){
    char expr[MAX];
    scanf("%s", expr);
    for(int i=0;expr[i]!='\0';i++){
        if(isdigit(expr[i])){
            push(expr[i]-'0');
        }
        else{
            float val2=pop();
            float val1=pop();
```

240701244

Scanned with OKEN Scanner

```
        switch(expr[i]){
            case'+':
            push(val1+val2);
            break;
            case '-':
            push(val1-val2);
            break;
            case'*':
            push(val1*val2);
            break;
            case '/':
            push(val1/val2);
            break;
        }
    }
    printf("%.0f",pop());
    return 0;
}
```

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

2. Problem Statement

Latha is taking a computer science course and has recently learned about infix and postfix expressions. She is fascinated by the idea of converting infix expressions into postfix notation. To practice this concept, she wants to implement a program that can perform the conversion for her.

Help Latha by designing a program that takes an infix expression as input and outputs its equivalent postfix notation.

Example

Input:

(3+4)5

Output:

34+5

Scanned with OKEN Scanner

Input Format

The input consists of a string, the infix expression to be converted to postfix notation.

Output Format

The output displays a string, the postfix expression equivalent of the input infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: A+B*C-D/E

Output: ABC*+DE/-

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define MAX 100
char stack[MAX];
int top=-1;
void push(char c){
    stack[++top]=c;
}
char pop(){
    return stack[top--];
}
int precedence(char op){
    if(op=='+'||op=='-')  return 1;
    if(op=='*'||op=='/')  return 2;
    return 0;
}
int isOperator(char c){
    return c=='+'||c=='-'||c=='*'||c=='/';
}
void infixtopostfix(char*infix,char*postfix){
```

Scanned with OKEN Scanner

```
    int i,j=0;
    char c;
    for(i=0;infix[i]!='\0';i++){
        c=infix[i];
        if(isalnum(c)){
            postfix[j++]=c;
        }
        else if(c=='('){
             push(c);
        }
        else if(c==')'){
            while(top!=-1&&stack[top]!='('){
                postfix[j++]=pop();
            }
            pop();
        }
        else if(isOperator(c)){
            while(top!=-1&&precedence(stack[top])>=precedence(c))
            {
                postfix[j++]=pop();
            }
            push(c);

        }
    }
    while(top!=-1){
        postfix[j++]=pop();
    }
    postfix[j]='\0';
}
int main(){
    char infix[MAX],postfix[MAX];
    scanf("%s",infix);
    infixtopostfix(infix,postfix);
    printf("%s",postfix);
    return 0;
}
```

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

3. Problem Statement

Scanned with OKEN Scanner

Rithi is building a simple text editor that allows users to type characters, undo their typing, and view the current text. She has implemented this text editor using an array-based stack data structure.

She has to develop a basic text editor with the following features:

Type a Character (Push): Users can type a character and add it to the text editor.Undo Typing (Pop): Users can undo their typing by removing the last character they entered from the editor.View Current Text (Display): Users can view the current text in the editor, which is the sequence of characters in the buffer.Exit: Users can exit the text editor application.

Write a program that simulates this text editor's undo feature using a character stack and implements the push, pop and display operations accordingly.

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, print: "Typed character: <character>" where <character> is the character that was pushed to the stack.
2. If the choice is 2, print: "Undo: Removed character <character>" where <character> is the character that was removed from the stack.
3. If the choice is 2, and if the stack is empty without any characters, print "Text editor buffer is empty. Nothing to undo."
4. If the choice is 3, print: "Current text: <character1> <character2> ...

Scanned with OKEN Scanner

<characterN>" where <character1>, <character2>, ... are the characters in the stack, starting from the last pushed character.
5. If the choice is 3, and there are no characters in the stack, print "Text editor buffer 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: 1 H
1 A
3
4

Output: Typed character: H
Typed character: A
Current text: A H

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100
char stack[MAX];
int top=-1;
void push(char c){
    if(top==MAX-1){
        printf("Stack overflow. Cannot type more characters.\n");
        return;
    }
    stack[++top]=c;
    printf("Typed character: %c\n",c);

}
void pop(){
    if(top==-1){
        printf("Text editor buffer is empty. Nothing to undo.\n");
        return;
```

Scanned with OKEN Scanner

```
    }
    char removedchar=stack[top--];
    printf("Undo: Removed character %c\n",removedchar);
}
void display(){
    if(top==-1){
        printf("Text editor buffer is empty.\n");
        return;
    }
    printf("Current text:");
    for(int i=top;i>=0;i--){
        printf(" %c",stack[i]);
    }
    printf("\n");
}
int main()
{
    int choice;
    char ch;
    while(1){
        scanf("%d",&choice);
        switch(choice){
            case 1:
            getchar();
            scanf("%c",&ch);
            push(ch);
            break;
            case 2:
            pop();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(0);
            default:
            printf("Invalid choice\n");
        }
    }
    return 0;
}
```

240701244

Scanned with OKEN Scanner

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

1. 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();
```

Scanned with OKEN Scanner

```
        }
        cur.setNext(temp);
    }
    size++;
}
```

Answer

Insert at the rear end of the dequeue

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

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

Answer

Enqueue and Dequeue

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

3. 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

10 30 40 15

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

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

Scanned with OKEN Scanner

```
#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***

5. What are the applications of dequeue?

Answer

All the mentioned options

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

Scanned with OKEN Scanner

6. 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***

7. 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***

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

Answer

public Object deleteFront() throws emptyDEQException{if(isEmpty())throw new emptyDEQException("Empty");else{Node temp = head.getNext();Node cur = temp.getNext();Object e = temp.getEle();head.setNext(cur);size--;return e;}}

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

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

Answer

First In First Out

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

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

Answer

Scanned with OKEN Scanner

All of the mentioned options

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

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

Answer

Queue

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

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

Answer

Overflow

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

13. 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***

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

Answer

Dequeue

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

15. What does the front pointer in a linked list implementation of a queue

Scanned with OKEN Scanner

contain?

Answer

The address of the first element

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

16. 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***

17. 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***

18. 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;
```

Scanned with OKEN Scanner

```
    }
    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***

19. What will the output of the following code?

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

240701244

Scanned with OKEN Scanner

```
#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***

20. 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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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."

240701244

Scanned with OKEN Scanner

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
```

Scanned with OKEN Scanner

```
int isEmpty() {
    //Type your code here
    if (front==-1)
        return 1;
    else
        return 0;
}

int isFull() {
    //Type your code here
    if(rear==MAX_SIZE- 1)
        return 1;
    else
        return 0;
}

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

}

int dequeue() {
    //Type your code here
    if(isEmpty())
    {
        printf("No orders in the queue.\n");
        return -1;
    }
```

240701244

Scanned with OKEN Scanner

```
    else
    {
        printf("Dequeued Order: %c\n",orders[front]);
        if(front==rear)
            front=rear=-1;
        else
            front=front+1;
        return 0;
    }
}

void display() {
    //Type your codlee here
    if(isEmpty())
    {
        printf("Queue is empty. No orders available.\n");
    }else{

        printf("Orders in the queue are:");
        for(int i=front;i<=rear;i++){
            printf(" %c",orders[i]);
        }
        printf("\n");
    }
}

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:
```

240701244

Scanned with OKEN Scanner

```
            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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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"

240701244

Scanned with OKEN Scanner

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;
}
```

Scanned with OKEN Scanner

```
// You are using GCC
int isEmpty() {
    //type your code here
    return (front==-1);
}

int isFull() {
    //type your code here
    return (rear==MAX_SIZE-1);
}

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

     }
}

int dequeue() {
    //type your code here
    if(isEmpty()){
      // printf("Queue is empty.\n");
      return 0;
    }
    else
    {
      //printf("Dequeued Helpdesk Ticket ID:%d",ticketIDs[front]);
      lastDequeued=ticketIDs[front];
      if(front==rear){
        front=rear=-1;
      }else{
        front=front+1;
```

Scanned with OKEN Scanner

```
        }
        return 1;
    }

}

void display() {
    //type your code here
    if(isEmpty()){
        printf("Queue is empty.\n");
    }
    else
    {
        printf("Helpdesk Ticket IDs in the queue are:");
        for(int i=front;i<=rear;i++){
            printf("%d\t",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;
```

240701244

Scanned with OKEN Scanner

```
        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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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.

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

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;

// You are using GCC
int insertq(int *data)
{
    //Type your code here
    if(rear==max- 1){
        return 0;
    }
    if(front==-1)
        front=0;
    rear=rear+1;
    queue[rear]=*data;
    return 1;


}

int delq()
{
    //Type your code here
    if(front==-1)
    {
        printf("Queue is empty.\n");
        return -1;
    }
    else
    {
        printf("Deleted number is: %d\n",queue[front]);
```

Scanned with OKEN Scanner

```
        if(front==rear)
            front=rear=-1;
        else
            front=front+1;
        return 0;

    }
}

void display()
{
    //Type your code here
    int i;
    if (front==-1){
        printf("Queue is empty.\n");
    }
    else
    {
        printf("Elements in the queue are:");
        for (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);
```

240701244

Scanned with OKEN Scanner

```
            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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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"

240701244

Scanned with OKEN Scanner

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

```
// You are using GCC
#include<stdio.h>
#define MAX_SIZE 5
int queue[MAX_SIZE];
int front=-1,rear=-1;
int isempty(){
```

240701244

Scanned with OKEN Scanner

```
    return front==-1;
}
int isfull(){
    return rear==MAX_SIZE-1;
}
void enqueue(int value){
    if(isfull()){
        printf("Queue is full. Cannot enqueue.\n");
    }
    else{
        if(front==-1)
            front=0;
        rear=rear+1;
        queue[rear]=value;
        printf("Print job with %d pages is enqueued.\n",queue[rear]);
        return;
    }
}
void dequeue(){
    if (isempty()){
        printf("Queue is empty.\n");
    }
    else{
        printf("Processing print job: %d pages\n",queue[front]);
        if(front==rear){
            front=rear=-1;
        }
        else{
            front=front+1;
        }
    }
}

void display(){
    if(isempty()){
        printf("Queue is empty.\n");
    }
    else{
        printf("Print jobs in the queue: ");
        for (int i=front;i<=rear;i++){
            printf("%d ",queue[i]);
        }
```

240701244

Scanned with OKEN Scanner

```
        printf("\n");
    }
}
int main(){
    int ch,p;
    while(1){
        scanf("%d",&ch);
        switch(ch){
            case 1:
            scanf("%d",&p);
            enqueue(p);
            break;
            case 2:
            dequeue();
            break;
            case 3:
            display();
            break;
            case 4:
            printf("Exiting program\n");
            return 0;
            default:
            printf("Invalid option.\n");
        }
    }
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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.

Scanned with OKEN Scanner

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;

// You are using GCC
void enqueue(int d) {
   //Type your code here
   struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
```

Scanned with OKEN Scanner

```
    newNode->data=d;
    newNode->next=NULL;
    if(rear==NULL){
        front=rear=newNode;
    }
    else{
        rear->next=newNode;
        rear=newNode;
    }

}

void printFrontRear() {
    //Type your code here
    if(front==NULL){
        printf("Queue is empty\n");
    }else{
        printf("Front: %d,Rear: %d\n",front->data,rear->data);
    }

}

void dequeue() {
    //Type your code here
    if(front==NULL){
        return;
    }
    struct Node*temp=front;
    front=front->next;
    if(front==NULL){
        rear=NULL;
    }
    free(temp);
}

int main() {
    int n, data;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(data);
    }
    printFrontRear();
```

240701244

Scanned with OKEN Scanner

```
    printf("Performing Dequeue Operation:\n");
    dequeue();
    printFrontRear();
    return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Sharon is developing a queue using an array. She wants to provide the functionality to find the Kth largest element. The queue should support the addition and retrieval of the Kth largest element effectively. The maximum capacity of the queue is 10.

Assist her in the program.

Input Format

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

The second line consists of N space-separated integers.

The third line consists of an integer K.

Scanned with OKEN Scanner

Output Format

For each enqueued element, print a message: "Enqueued: " followed by the element.

The last line prints "The [K]th largest element: " followed by the Kth largest element.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 45 93 87 25
4
Output: Enqueued: 23
Enqueued: 45
Enqueued: 93
Enqueued: 87
Enqueued: 25
The 4th largest element: 25

Answer

```
// You are using GCC
#include<stdio.h>
void sortdes(int arr[],int n){
   int i,j,temp;
   for (i=0;i<n-1;i++){
      for(j=i+1;j<n;j++){
         if(arr[i]<arr[j]){
            temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
         }
      }
   }
}
int main(){
   int queue[10];
   int n,k;
```

240701244

Scanned with OKEN Scanner

```
    scanf("%d",&n);
    if(n>10){
        printf("Queue can hold");
        return 1;
    }
    for(int i=0;i<n;i++){
        scanf("%d",&queue[i]);
        printf("Enqueued: %d\n",queue[i]);
    }
    scanf("%d",&k);
    sortdes(queue,n);
    if(k<=n){
        printf("The %dth largest element: %d\n",k,queue[k-1]);
    }else{
        printf("Cannot find the %dth largest element.",k);
    }
    return 0;
}
```

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

2. Problem Statement

You've been assigned the challenge of developing a queue data structure using a linked list.

The program should allow users to interact with the queue by enqueuing positive integers and subsequently dequeuing and displaying elements.

Input Format

The input consists of a series of integers, one per line. Enter positive integers into the queue.

Enter -1 to terminate input.

Output Format

The output prints the space-separated dequeued elements.

Scanned with OKEN Scanner

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

Sample Test Case

Input: 1
2
3
4
-1

Output: Dequeued elements: 1 2 3 4

Answer

```
// You are using GCC
#include<stdio.h>
#define MAX 1000
int main(){
    int queue[MAX];
    int front =0,rear=0;
    int num;
    while(1){
        scanf("%d",&num);
        if(num==-1){
            break;
        }
        if(num>0){
            queue[rear]=num;
            rear++;
        }
    }
    printf("Dequeued elements: ");
    while(front<rear){
        printf("%d ",queue[front]);
        front++;
    }
    printf("\n");
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

3. Problem Statement

Guide Harish in developing a simple queue system for a customer service center. The customer service center can handle up to 25 customers at a time. The queue needs to support basic operations such as adding a customer to the queue, serving a customer (removing them from the queue), and displaying the current queue of customers.

Use an array for implementation.

Input Format

The first line of the input consists of an integer N, the number of customers arriving at the service center.

The second line consists of N space-separated integers, representing the customer IDs in the order they arrive.

Output Format

After serving the first customer in the queue, display the remaining customers in the queue.

If a dequeue operation is attempted on an empty queue, display "Underflow".

If the queue is empty, display "Queue is empty".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 102 103 104 105
Output: 102 103 104 105

Answer

```
// You are using GCC
#include<stdio.h>
int main(){
   int n,i;
```

240701244

Scanned with OKEN Scanner

```
    int queue[25];
    scanf("%d",&n);
    if(n==0){
        ////int dummy;
        //scanf("%d",&dummy);
        printf("Underflow\n");
        printf("Queue is empty\n");
        return 0;
    }
    for (i=0;i<n;i++){
        scanf("%d",&queue[i]);
    }
    for(i=1;i<n;i++){
        printf("%d ",queue[i]);
    }
    //if(n==1){
      // printf("Queue is empty");
    //}
    return 0;
}
```

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

4. Problem Statement

Amar is working on a project where he needs to implement a special type of queue that allows selective dequeuing based on a given multiple. He wants to efficiently manage a queue of integers such that only elements not divisible by a given multiple are retained in the queue after a selective dequeue operation.

Implement a program to assist Amar in managing his selective queue.

Example

Input:

5

10 2 30 4 50

Scanned with OKEN Scanner

5

Output:

Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Explanation:

After selective dequeue with a multiple of 5, the elements that are multiples of 5 should be removed. Therefore, only 10, 30, and 50 should be removed from the queue. The updated Queue is 2 4.

Input Format

The first line contains an integer n, representing the number of elements initially present in the queue.

The second line contains n space-separated integers, representing the elements of the queue.

The third line contains an integer multiple, representing the divisor for selective dequeue operation.

Output Format

The first line of output prints "Original Queue: " followed by the space-separated elements in the queue before the dequeue operation.

The second line prints "Queue after selective dequeue: " followed by the remaining space-separated elements in the queue, after deleting elements that are the multiples of the specified number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
10 2 30 4 50
5
Output: Original Queue: 10 2 30 4 50
Queue after selective dequeue: 2 4

240701244

Scanned with OKEN Scanner

Answer

```
// You are using GCC
#include<stdio.h>
int main(){
    int n,i,multiple;
    int queue[50];
    int result[50];
    int resultsize=0;
    scanf("%d",&n);
    for(i=0;i<n;i++){
        scanf("%d",&queue[i]);
    }
    scanf("%d",&multiple);
    printf("Original Queue: ");
    for (i=0;i<n;i++){
        printf("%d ",queue[i]);
    }
    printf("\n");
    for(i=0;i<n;i++){
        if(queue[i]%multiple!=0){
            result[resultsize++]=queue[i];
        }
    }
    printf("Queue after selective dequeue: ");
    for(i=0;i<resultsize;i++){
        printf("%d ",result[i]);
    }
    printf("\n");
    return 0;
}
```

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

5. Problem Statement

You are tasked with developing a simple ticket management system for a customer support department. In this system, customers submit support tickets, which are processed in a First-In-First-Out (FIFO) order. The system needs to handle the following operations:

Scanned with OKEN Scanner

Ticket Submission (Enqueue Operation): New tickets are submitted by customers. Each ticket is assigned a unique identifier (represented by an integer). When a new ticket arrives, it should be added to the end of the queue.

Ticket Processing (Dequeue Operation): The support team processes tickets in the order they are received. The ticket at the front of the queue is processed first. After processing, the ticket is removed from the queue.

Display Ticket Queue: The system should be able to display the current state of the ticket queue, showing the sequence of ticket identifiers from front to rear.

Input Format

The first input line contains an integer n, the number of tickets submitted by customers.

The second line consists of a single integer, representing the unique identifier of each submitted ticket, separated by a space.

Output Format

The first line displays the "Queue: " followed by the ticket identifiers in the queue after all tickets have been submitted.

The second line displays the "Queue After Dequeue: " followed by the ticket identifiers in the queue after processing (removing) the ticket at the front.

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

Sample Test Case

Input: 6
14 52 63 95 68 49

Output: Queue: 14 52 63 95 68 49
Queue After Dequeue: 52 63 95 68 49

Answer

// You are using GCC

240701244

Scanned with OKEN Scanner

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int queue[n];
    for(int i=0;i<n;i++){
        scanf("%d",&queue[i]);
    }
    printf("Queue:");
    for(int i=0;i<n;i++)
    {
        printf("%d ",queue[i]);
    }
    printf("\n");
    printf("Queue After Dequeue: ");
    for (int i=1;i<n;i++){
        printf("%d ",queue[i]);
    }
    printf("\n");
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Saran is developing a simulation for a theme park where people wait in a queue for a popular ride.

Each person has a unique ticket number, and he needs to manage the queue using a linked list implementation.

Your task is to write a program for Saran that reads the number of people in the queue and their respective ticket numbers, enqueue them, and then calculate the sum of all ticket numbers to determine the total ticket value present in the queue.

Input Format

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

Scanned with OKEN Scanner

in the queue.

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

Output Format

The output prints an integer representing the sum of all ticket numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5

Output: 24

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node*next;
};
void enqueue(int data,struct node**front,struct node**rear){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->next=NULL;
   if(*rear==NULL){
      *front=*rear=newnode;
      return;
   }
   (*rear)->next=newnode;
   *rear=newnode;
}
int sum(struct node* front){
   int s=0;
   struct node*temp=front;
   while (temp!=NULL){
      s+=temp->data;
```

Scanned with OKEN Scanner

```
        temp=temp->next;
    }
    return s;
}
int main(){
    int n;
    scanf("%d",&n);
    struct node*front=NULL;
    struct node*rear=NULL;
    for(int i=0;i<n;i++){
        int data;
        scanf("%d",&data);
        enqueue(data,&front,&rear);
    }
    printf("%d\n",sum(front));
    return 0;
}
```

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

2. Problem Statement

Imagine you are developing a basic task management system for a small team of software developers. Each task is represented by an integer, where positive integers indicate valid tasks and negative integers indicate erroneous tasks that need to be removed from the queue before processing.

Write a program using the queue with a linked list that allows the team to add tasks to the queue, remove all erroneous tasks (negative integers), and then display the valid tasks that remain in the queue.

Input Format

The first line consists of an integer N, representing the number of tasks to be added to the queue.

The second line consists of N space-separated integers, representing the tasks. Tasks can be both positive (valid) and negative (erroneous).

Output Format

Scanned with OKEN Scanner

The output displays the following format:

For each task enqueued, print a message "Enqueued: " followed by the task value.

The last line displays the "Queue Elements after Dequeue: " followed by removing all erroneous (negative) tasks and printing the valid tasks remaining in the queue in the order they were enqueued.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
12 -54 68 -79 53

Output: Enqueued: 12
Enqueued: -54
Enqueued: 68
Enqueued: -79
Enqueued: 53
Queue Elements after Dequeue: 12 68 53

Answer

```
// You are using GCC
#include<stdio.h>
#define MAX 15
void processQueue(int queue[],int n){
    printf("Queue Elements after Dequeue: ");
    for(int i=0;i<n;i++){
        if(queue[i]>=0){
            printf("%d ",queue[i]);
        }
    }
    printf("\n");
}
int main(){
    int N;
    int queue[MAX];
    scanf("%d",&N);
    if(N<1||N>MAX){
```

240701244

Scanned with OKEN Scanner

```
        printf("Invalid.");
        return 1;
    }
    for (int i=0;i<N;i++){
        scanf("%d",&queue[i]);
        printf("Enqueued: %d\n",queue[i]);
    }
    processQueue(queue, N);
    return 0;
}
```

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

3. Problem Statement

Fathima has been tasked with developing a program to manage a queue of customers waiting in line at a service center. Help her write a program simulating a queue data structure using a linked list.

Here is a description of the scenario and the required operations:

Enqueue: Add a customer to the end of the queue.Dequeue: Remove and discard a customer from the front of the queue.Display waiting customers: Display the front and rear customer IDs in the queue.

Write a program that enqueues all the customers into the queue, performs a dequeue operation, and prints the front and rear elements.

Input Format

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

The second line consists of N space-separated integers, representing the customer IDs.

Output Format

The output prints "Front: X, Rear: Y" where X is the front element and Y is the rear element, after performing the dequeue operation.

Scanned with OKEN Scanner

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

Sample Test Case

Input: 5
112 104 107 116 109

Output: Front: 104, Rear: 109

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node*next;
};
void enqueue(struct node**front,struct node**rear,int value){
    struct node*newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=value;
    newnode->next=NULL;
    if(*rear==NULL){
        *front=*rear=newnode;
    }else{
        (*rear)->next=newnode;
        *rear=newnode;
    }
}
void dequeue(struct node**front,struct node**rear){
    if(*front==NULL){
        return;
    }
    struct node*temp=*front;
    *front=(*front)->next;
    if(*front==NULL){
        *rear=NULL;
    }
    free(temp);
}
void display(struct node*front,struct node*rear){
    if(front!=NULL && rear!=NULL){
        printf("Front: %d, Rear: %d\n",front->data,rear->data);
```

Scanned with OKEN Scanner

```
    }else{
        printf("Empty");
    }
}
int main(){
    int n,value;
    scanf("%d",&n);
    struct node*front=NULL;
    struct node*rear=NULL;
    if(n<2||n>20){
        printf("Invalid");
        return 1;
    }
    for(int i=0;i<n;i++){
        scanf("%d",&value);
        enqueue(&front, &rear, value);

    }
    dequeue(&front,&rear);
    display(front, rear);
    return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_MCQ

Attempt : 2
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. 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***

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

Answer

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

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

Scanned with OKEN Scanner

3. 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***

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

Answer

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

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

5. 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***

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

Answer

Scanned with OKEN Scanner

12

Status : Correct *Marks : 1/1*

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

Answer

1, 4, 2, 18, 14, 13

Status : Correct *Marks : 1/1*

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

Answer

67

Status : Correct *Marks : 1/1*

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

Answer

13, 2, 1, 4, 14, 18

Status : Correct *Marks : 1/1*

11. 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*

Scanned with OKEN Scanner

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

Answer

1, 2, 4, 13, 14, 18

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

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

Answer

14

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

14. 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***

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

Answer

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

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 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.

Scanned with OKEN Scanner

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;
}

// You are using GCC
struct TreeNode* insert(struct TreeNode* root, int key) {
   //Type your code here
```

Scanned with OKEN Scanner

```
    if(root==NULL) return createNode(key);
    if(key < root->data)
        root->left=insert(root->left,key);
    else if(key > root->data)
        root->right=insert(root->right,key);

    return root;
}

struct TreeNode* findMin(struct TreeNode* root) {
    //Type your code here
    while (root && root->left!=NULL){
        root=root->left;
    }
    return root;
}

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

240701244

Scanned with OKEN Scanner

```
void inorderTraversal(struct TreeNode* root) {
    //Type your code here
    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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

Write a program to help him solve this program.

Input Format

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

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

Output Format

Scanned with OKEN Scanner

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 1 5 2 4

Output: 3 1 2 5 4

Answer

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

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

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

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

void printPreorder(struct Node* node) {
```

Scanned with OKEN Scanner

```
    //Type your code here
    if(node==NULL){
        return;
    }
    printf("%d ",node->data);
    printPreorder(node->left);
    printPreorder(node->right);

}

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

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

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

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

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 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

Scanned with OKEN Scanner

in the binary search tree.

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 3 10 1 6 14 23
6

Output: Value 6 is found in the tree.

Answer

```
// You are using GCC

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

   }
```

Scanned with OKEN Scanner

```
    else if(value>root->data){
        return searchNode(root->right,value);
    }
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

Help him by implementing the program.

Input Format

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

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

Scanned with OKEN Scanner

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

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

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

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

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

Scanned with OKEN Scanner

```
    return root;
}

void displayTreePostOrder(struct Node* root) {
    //Type your code here
    if(root ==NULL)   return;
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d ",root->data);
}
int findMinValue(struct Node* root) {
    //Type your code here
    if(root==NULL)
        return -1;
    while (root->left!=NULL){
        root=root->left;
    }
    return root->data;
}

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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

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

Input Format

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

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

Output Format

Scanned with OKEN Scanner

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7

Output: 15

Answer

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

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

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

int findMax(struct TreeNode* root) {
   //Type your code here
```

Scanned with OKEN Scanner

```
    if(root==NULL)
        return -1;
    while(root->right!=NULL){
        root=root->right;
    }
    return root->data;
}

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Joseph, a computer science student, is interested in understanding binary search trees (BST) and their node arrangements. He wants to create a program to explore BSTs by inserting elements into a tree and displaying the nodes using post-order traversal of the tree.

Write a program to help Joseph implement 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.

Scanned with OKEN Scanner

Output Format

The output prints N space-separated integer values after the post-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 15 5 3
Output: 3 5 15 10

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
   int value;
   struct node*left,*right;
};
struct node*create(int value){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->value=value;
   newnode->left=NULL;
   newnode->right=NULL;
   return newnode;
}
struct node*insert(struct node*root,int value){
   if(root==NULL)    return create(value);
   if(value<root->value)
   root->left=insert(root->left,value);
   else if(value>root->value)
   root->right=insert(root->right,value);
   return root;
}
void postorder(struct node*root){
   if(root==NULL)  return;
   postorder(root->left);
   postorder(root->right);
```

240701244

Scanned with OKEN Scanner

```
    printf("%d ",root->value);
}
int main()
{
    int n;
    scanf("%d",&n);
    struct node*root=NULL;
    int m;
    for(int i=0;i<n;i++){
        scanf("%d",&m);
        root=insert(root,m);
    }
    postorder(root);
    return 0;
}
```

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

2. Problem Statement

Aishu is participating in a coding challenge where she needs to reconstruct a Binary Search Tree (BST) from given preorder traversal data and then print the in-order traversal of the reconstructed BST.

Since Aishu is just learning about tree data structures, she needs your help to write a program that does this efficiently.

Input Format

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

The second line of input contains n integers separated by spaces, which represent the preorder traversal of the BST.

Output Format

The output displays n space-separated integers, representing the in-order traversal of the reconstructed BST.

Scanned with OKEN Scanner

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
10 5 1 7 40 50

Output: 1 5 7 10 40 50

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node*left,*right;

};
struct node*create(int data){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->left=NULL;
   newnode->right=NULL;
   return newnode;
}
struct node*insert(struct node*root,int data){
   if(root==NULL){
      return create(data);
   }
   if(data<root->data)
   root->left=insert(root->left,data);
   else if(data>root->data)
   root->right=insert(root->right,data);
   return root;
}
void inorder(struct node*root){
   if(root==NULL)   return;
   inorder(root->left);
   printf("%d ",root->data);
   inorder(root->right);

}
int main()
```

Scanned with OKEN Scanner

```
{
    int n;
    int m;
    scanf("%d",&n);
    struct node*root=NULL;
    for(int i=0;i<n;i++){
        scanf("%d",&m);
        root=insert(root,m);
    }
    inorder(root);
    return 0;
}
```

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

3. Problem Statement

Yogi is working on a program to manage a binary search tree (BST) containing integer values. He wants to implement a function that removes nodes from the tree that fall outside a specified range defined by a minimum and maximum value.

Help Yogi by writing a function that achieves this.

Input Format

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

The second line consists of N space-separated integers, representing the elements to be inserted into the BST.

The third line consists of two space-separated integers min and max, representing the minimum value and the maximum value of the range.

Output Format

The output prints the remaining elements of the BST in an in-order traversal, after removing nodes that fall outside the specified range.

240701244

Scanned with OKEN Scanner

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 12
5 15

Output: 5 10 12 15

Answer

```
// You are using
#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=NULL;
   newNode->right=NULL;
   return newNode;
}
struct Node*insert(struct Node*root,int data){
   if(root==NULL) {
      return createNode(data);
   }
   if(data<root->data){
      root->left=insert(root->left,data);
   }
   else{
      root->right=insert(root->right,data);
   }
   return root;
}
void inorder(struct Node*root){
   if(root==NULL)
      return;
   inorder(root->left);
   printf("%d ",root->data);
```

Scanned with OKEN Scanner

```
    inorder(root->right);
}
struct Node*removeOutsideRange(struct Node*root,int min,int max){
    if(root==NULL)
        return NULL;
    root->left=removeOutsideRange(root->left,min,max);
    root->right=removeOutsideRange(root->right,min,max);
    if(root->data < min){
        struct Node* rightChild=root->right;
        free(root);
        return rightChild;
    }
    if(root->data > max){
        struct Node* leftChild=root->left;
        free(root);
        return leftChild;
    }
    return root;
}
int main(){
    int n,data;
    scanf("%d",&n);
    struct Node*root=NULL;
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        root=insert(root,data);
    }
    int min,max;
    scanf("%d %d",&min,&max);
    root=removeOutsideRange(root,min,max);
    inorder(root);
    return 0;
}
```

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

4. Problem Statement

Arun is exploring operations on binary search trees (BST). He wants to write a program with an unsorted distinct integer array that represents the BST keys and construct a height-balanced BST from it.

Scanned with OKEN Scanner

After constructing, he wants to perform the following operations that can alter the structure of the tree and traverse them using a level-order traversal:

InsertionDeletion

Your task is to assist Arun in completing the program without any errors.

Input Format

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

The second line consists of N space-separated integers, representing the initial keys.

The third line consists of an integer X, representing the new key to be inserted into the BST.

The fourth line consists of an integer Y, representing the key to be deleted from the BST.

Output Format

The first line of output prints "Initial BST: " followed by a space-separated list of keys in the initial BST after constructing it in level order traversal.

The second line prints "BST after inserting a new node X: " followed by a space-separated list of keys in the BST after inserting X n level order traversal.

The third line prints "BST after deleting node Y: " followed by a space-separated list of keys in the BST after deleting Y n level order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
25 14 56 28 12
34
12

Scanned with OKEN Scanner

Output: Initial BST: 25 14 56 12 28
BST after inserting a new node 34: 25 14 56 12 28 34
BST after deleting node 12: 25 14 56 28 34

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node*left,*right;

};
struct node*create(int data){
   struct node*node=(struct node*)malloc(sizeof(struct node));
   node->data=data;
   node->left=NULL;
   node->right=NULL;
   return node;
}
struct node*insert(struct node*root,int data){
   if(root==NULL)  return create(data);
   if(data<root->data)
   root->left=insert(root->left,data);
   else if(data>root->data)
   root->right=insert(root->right,data);
   return root;
}
struct node*findmin(struct node*root){
   while(root->left!=NULL)
      root=root->left;
   return root;
}
struct node*deletenode(struct node*root,int key){
   if(root==NULL)  return NULL;
   if(key<root->data)
   root->left=deletenode(root->left,key);
   else if (key>root->data)
   root->right=deletenode(root->right,key);
   else{
      if(root->left==NULL){
         struct node*temp=root->right;
```

Scanned with OKEN Scanner

```
        free(root);
        return temp;
    }
    if(root->right==NULL){
        struct node*temp=root->left;
        free(root);
        return temp;
    }
    struct node*temp=findmin(root->right);
    root->data=temp->data;
    root->right=deletenode(root->right,temp->data);
}
return root;
}
void levelorder(struct node*root){
    if(root==NULL)
        return;
    struct node*queue[100];
    int front=0,rear=0;
    queue[rear++]=root;
    while(front<rear){
        struct node*curr=queue[front++];
        printf("%d ",curr->data);
        if(curr->left)
            queue[rear++]=curr->left;
        if(curr->right)
            queue[rear++]=curr->right;
    }
}
int main(){
    int n,arr[20];
    scanf("%d",&n);
    for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
    int x,y;
    scanf("%d %d",&x,&y);
    struct node*root=NULL;
    for(int i=0;i<n;i++)
        root=insert(root,arr[i]);
    printf("Initial BST: ");
    levelorder(root);
    printf("\n");
```

240701244

Scanned with OKEN Scanner

```
    root=insert(root,x);
    printf("BST after inserting a new node %d: ",x);
    levelorder(root);
    printf("\n");
    root=deletenode(root,y);
    printf("BST after deleting node %d: ",y);
    levelorder(root);
    printf("\n");
    return 0;

}
```

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

5. Problem Statement

Viha, a software developer, is working on a project to automate searching for a target value in a Binary Search Tree (BST). She needs to create a program that takes an integer target value as input and determines if that value is present in the BST or not.

Write a program to assist Viha.

Input Format

The first line of input consists of integers separated by spaces, which represent the elements to be inserted into the BST. The input is terminated by entering -1.

The second line consists of an integer target, which represents the target value to be searched in the BST.

Output Format

If the target value is found in the BST, print "[target] is found in the BST".

Else, print "[target] is not found in the BST"

Refer to the sample output for formatting specifications.

Scanned with OKEN Scanner

Sample Test Case

Input: 5 3 7 1 4 6 8 -1
4
Output: 4 is found in the BST

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node*left,*right;
};
struct node*create(int data){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->left=NULL;
   newnode->right=NULL;
   return newnode;
}
struct node*insert(struct node*root,int data){
   if(root==NULL){
      return create(data);
   }
   if(data<root->data)
   root->left=insert(root->left,data);
   else if(data>root->data)
   root->right=insert(root->right,data);
   return root;

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

Scanned with OKEN Scanner

```
    struct node *root=NULL;
    int data;
    while(1){
        scanf("%d",&data);
        if(data==-1)
            break;
        root=insert(root,data);
    }
    int target;
    scanf("%d",&target);
    if(search(root,target))
    printf("%d is found in the BST",target);
    else
    printf("%d is not found in the BST",target);
    return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Emily is studying binary search trees (BST). She wants to write a program that inserts characters into a BST and then finds and prints the minimum and maximum values.

Guide her with the program.

Input Format

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

The second line consists of N space-separated characters.

Output Format

The first line of output prints "Minimum value: " followed by the minimum value

Scanned with OKEN Scanner

of the given inputs.

The second line prints "Maximum value: " followed by the maximum value of the given inputs.

Refer to the sample outputs for formatting specifications.

Sample Test Case

Input: 5
Z E W T Y
Output: Minimum value: E
Maximum value: Z

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
   char value;
   struct node*left;
   struct node*right;
};
struct node*create(char value){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->value=value;
   newnode->left=NULL;
   newnode->right=NULL;
   return newnode;
}
struct node*insert(struct node*root,int value){
   if(root==NULL)  return create(value);
   if(value<root->value)
   root->left=insert(root->left,value);
   else if(value>root->value)
   root->right=insert(root->right,value);
   return root;
}
char findmin(struct node*root){
   if(root==NULL){
```

Scanned with OKEN Scanner

```
        return -1;
    }
    while(root->left!=NULL){
        root=root->left;
    }
    return root->value;
}
char findmax(struct node*root){
    if(root==NULL){
        return -1;
    }
    while(root->right!=NULL){
        root=root->right;
    }
    return root->value;
}
int main()
{
    int n;
    char m;
    struct node*root=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf(" %c",&m);
        root=insert(root, m);
    }
    printf("Minimum value: %c\n",findmin(root));
    printf("Maximum value: %c",findmax(root));
    return 0;
}
```

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

2. Problem Statement

Arun is working on a Binary Search Tree (BST) data structure. His goal is to implement a program that reads a series of integers and inserts them into a BST. Once the integers are inserted, he needs to add a given integer value to each node in the tree and find the maximum value in the BST.

Scanned with OKEN Scanner

Your task is to help Arun implement this program.

Input Format

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

The second line consists of N space-separated integers, each representing an element to be inserted into the BST.

The third line consists of an integer add, representing the value to be added to each node in the BST.

Output Format

The output prints the maximum value in the BST after adding the add value.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 25
5

Output: 30

Answer

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

Scanned with OKEN Scanner

```
struct node*insert(struct node*root,int value){
    if(root==NULL)    return create(value);
    if(value<root->value)
    root->left=insert(root->left,value);
    else if(value>root->value)
    root->right=insert(root->right,value);
    return root;
}
void addto(struct node*root,int add){
    if(root==NULL)   return;
    root->value+=add;
    addto(root->left,add);
    addto(root->right,add);
}
int findmax(struct node*root){
    if(root==NULL)  return -1;
    while(root->right!=NULL){
        root=root->right;
    }
    return root->value;
}
int main(){
    int n,value,add;
    struct node*root=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&value);
        root=insert(root,value);
    }
    scanf("%d",&add);
    addto(root,add);
    printf("%d\n",findmax(root));
    return 0;
}
```

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

3. Problem Statement

Jake is learning about binary search trees(BST) and their operations. He wants to implement a program that can delete a node from a BST based

Scanned with OKEN Scanner

on the given key value and print the remaining nodes in an in-order traversal.

Assist Jake in the program.

Input Format

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

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

The third line consists of an integer x, representing the key value of the node to be deleted.

Output Format

The first line of output prints "Before deletion: " followed by the in-order traversal of the initial BST.

The second line prints "After deletion: " followed by the in-order traversal after the deletion of the key value.

If the key value is not present in the BST, print the original tree as it is.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 6 4 3 1
4

Output: Before deletion: 1 3 4 6 8
After deletion: 1 3 6 8

Answer

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

240701244

Scanned with OKEN Scanner

```
    int value;
    struct node*left,*right;
};
struct node*create(int value){
    struct node*newnode=(struct node*)malloc(sizeof(struct node));
    newnode->value=value;
    newnode->left=NULL;
    newnode->right=NULL;
    return newnode;
}
struct node*insert(struct node*root,int value){
    if(root==NULL)  return create(value);
    if(value<root->value)
    root->left=insert(root->left,value);
    else if(value>root->value)
    root->right=insert(root->right,value);
    return root;
}
void inorder(struct node* root){
    if(root==NULL)  return;
    inorder(root->left);
    printf("%d ",root->value);
    inorder(root->right);
}
struct node*findmin(struct node*root){
    while(root->left!=NULL)
        root=root->left;
    return root;
}
struct node*deleten(struct node*root,int key){
    if(root==NULL) return root;
    if(key<root->value)
    root->left=deleten(root->left,key);
    else if(key>root->value)
    root->right=deleten(root->right,key);
    else{
        if(root->left==NULL){
            struct node*temp=root->right;
            free(root);
            return temp;
        }else if(root->right==NULL){
            struct node*temp=root->left;
```

240701244

Scanned with OKEN Scanner

```
        free(root);
        return temp;
    }
    struct node*temp=findmin(root->right);
    root->value=temp->value;
    root->right=deleten(root->right,temp->value);
  }
  return root;
}
int search(struct node*root,int key){
  if(root==NULL)  return 0;
  if(key==root->value) return 1;
  if(key < root->value)  return search(root->left,key);
  else return search(root->right,key);
}
int main(){
  int n,value,x;
  struct node*root=NULL;
  scanf("%d",&n);
  for(int i=0;i<n;i++){
    scanf("%d",&value);
    root=insert(root,value);
  }
  scanf("%d",&x);
  printf("Before deletion: ");
  inorder(root);
  printf("\n");
  if(search(root, x)){
    root=deleten(root,x);
  }
  printf("After deletion: ");
  inorder(root);
  printf("\n");
  return 0;
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_MCQ_Updated_1

Attempt : 2
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

1. 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***

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

Answer

It requires additional memory for merging

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

Scanned with OKEN Scanner

3. 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***

4. Which of the following is true about Quicksort?

Answer

It is an in-place sorting algorithm

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

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

Answer

Two sorted subarrays are combined into one sorted array

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

6. 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***

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

Answer

merging

Scanned with OKEN Scanner

Status : Correct *Marks : 1/1*

8. 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*

9. 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*

10. Merge sort is _____.

Answer

Comparison-based sorting algorithm

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 left of the pivot

Status : Correct *Marks : 1/1*

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

Scanned with OKEN Scanner

Answer

Switching to Insertion Sort for small subarrays

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

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

Answer

It is used to partition the array

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

14. Is Merge Sort a stable sorting algorithm?

Answer

Yes, always stable.

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

15. 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***

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

Answer

Quicksort requires less auxiliary space

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

Scanned with OKEN Scanner

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

Answer

Merge Sort

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 scenarios is Merge Sort preferred over Quick Sort?

Answer

When sorting linked lists

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

20. 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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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).

Scanned with OKEN Scanner

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) {

    //Type your code here
    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++];
```

Scanned with OKEN Scanner

```
    }
}

void mergeSort(int arr[], int size) {
    //Type your code here
    if (size<2)   return;
    int mid=size/2;
    int left[mid],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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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>

// You are using GCC

void insertionSort(int arr[], int n) {
    //Type your code here
    for(int i=1;i<n;i++){
        int key;
        key=arr[i];
        int j=i-1;
        while (j>=0 && arr[j]>key){
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=key;
    }
}

void printArray(int arr[], int n) {
    //Type your code here
    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++) {
```

Scanned with OKEN Scanner

```
        scanf("%d", &arr[i]);
    }

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

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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.

Scanned with OKEN Scanner

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>

// You are using GCC
void swap(char* a, char* b) {
    //Type your code here
    int t=*a;
    *a=*b;
    *b=t;
}

int partition(char arr[], int low, int high) {
    //Type your code here
    int pivot=arr[low];
    int i=low+1;
    int j=high;
    while(i<=j)
    {
        while(i<=high && arr[i]>pivot) i++;
        while(j>=low && arr[j]<pivot) j--;
        if(i<=j){
            swap(&arr[i],&arr[j]);
            i++;
```

Scanned with OKEN Scanner

```
            j--;
        }

    }
    swap(&arr[low],&arr[j]);
    return j;
}

void quicksort(char arr[], int low, int high) {
    //Type your code here
    if(low<high){
        int pi=partition(arr, low, high);
        quicksort(arr, low, pi-1);
        quicksort(arr, 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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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>

// You are using GCC
void swap(int *a,int *b)
{
    int t=*a;
    *a=*b;
    *b=t;
}
int partition(int arr[], int low, int high) {
    //Type your code here
    int pivot=arr[low];
    int i=low+1;
    int j=high;
    while(i<=j){

        while(i<=high && arr[i]>pivot){
            i++;
        }
        while(j>low && arr[j]<=pivot)
        {
            j--;
        }
```

Scanned with OKEN Scanner

```
        if(i<j)
        {
            swap(&arr[i],&arr[j]);
            i++;
            j--;
        }
    }
    swap(&arr[low],&arr[j]);
    return j;
}


void quickSort(int arr[], int low, int high) {
    //Type your code here
    if(low<high){
        int p=partition(arr, low, high);
        quickSort(arr, low, p-1);
        quickSort(arr, p+1, high);
    }
}

int findNthLargest(int* nums, int n, int k) {
    //Type your code here
    quickSort(nums, 0, n-1);
    printf("%d",nums[k-1]);
    return 0;
}
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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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>

// You are using GCC
int compare(double a, double b) {
    //Type your code here
    return a>b;
}
void merge(double arr[], int l, int m, int r) {
    //Type your code here
    int n1=m-l+1,n2=r-m;
    double L[n1],R[n2];
    for (int i=0;i<n1;i++)
        L[i]=arr[l+i];
    for(int i=0;i<n2;i++)
        R[i]=arr[m+1+i];
    int i=0,j=0,k=l;
    while(i<n1 && j<n2){
        if(!compare(L[i],R[j]))
            arr[k++]=L[i++];
        else
            arr[k++]=R[j++];
    }
    while(i<n1)
        arr[k++]=L[i++];
    while(j<n2)
        arr[k++]=R[j++];
```

240701244

Scanned with OKEN Scanner

```
}
void mergeSort(double arr[], int l, int r) {
    //Type your code here
    if(l<r){
        int m=(l+r)/2;
        mergeSort(arr,l,m);
        mergeSort(arr,m+1,r);
        merge(arr, l, m, r);
    }
}

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**

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

You are working as a programmer at a sports academy, and the academy holds various sports competitions regularly.

As part of the academy's system, you need to sort the scores of the participants in descending order using the Quick Sort algorithm.

Write a program that takes the scores of n participants as input and uses the Quick Sort algorithm to sort the scores in descending order. Your program should display the sorted scores after the sorting process.

Input Format

The first line of input consists of an integer n, which represents the number of scores.

Scanned with OKEN Scanner

The second line of input consists of n integers, which represent scores separated by spaces.

Output Format

Each line of output represents an iteration of the Quick Sort algorithm, displaying the elements of the array at that iteration.

After the iterations are complete, the last line of output prints the sorted scores in descending order separated by space.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
78 54 96 32 53

Output: Iteration 1: 78 54 96 53 32
Iteration 2: 96 54 78
Iteration 3: 78 54
Sorted Order: 96 78 54 53 32

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
void swap(int *a,int *b)
{
    int t=*a;
    *a=*b;
    *b=t;
}
int pivot(int arr[],int low,int high){
    int pi=arr[high];
    int i=low-1;
    int j=0;
    for(j=low;j<=high-1;j++){
        if(arr[j]>=pi){
            i++;
            swap(&arr[i],&arr[j]);
```

Scanned with OKEN Scanner

```
        }
    }
    swap(&arr[i+1],&arr[j]);
    return (i+1);
}
int itr=1;
void quicksort(int arr[],int l,int h,int n){
    if(l<h){
        int p=pivot(arr,l,h);
        printf("Iteration %d: ",itr++);
        for(int i=l;i<=h;i++){
            printf("%d ",arr[i]);
        }
        printf("\n");
        quicksort(arr,l,p-1,n);
        quicksort(arr,p+1,h,n);
    }
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    quicksort(a,0,n-1,n);
    printf("Sorted Order: ");
    for(int i=0;i<n;i++){
        printf("%d ",a[i]);

    }

}
```

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

2. Problem Statement

Alex is working on a project that involves merging and sorting two arrays. He wants to write a program that merges two arrays, sorts the merged array in ascending order, removes duplicates, and prints the sorted array

Scanned with OKEN Scanner

without duplicates.

Help Alex to implement the program using the merge sort algorithm.

Input Format

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

The second line consists of N integers, separated by spaces, representing the elements of the first array.

The third line consists of an integer M, representing the number of elements in the second array.

The fourth line consists of M integers, separated by spaces, representing the elements of the second array.

Output Format

The output prints space-separated integers, representing the merged and sorted array in ascending order, with duplicate elements removed.

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
// You are using GCC
#include<stdio.h>
void merge(int a[],int l,int m,int r)
{
    int i=l,j=m+1,k=0;
    int b[r-l+1];
    while(i<=m && j<=r){
```

240701244

Scanned with OKEN Scanner

```
        if(a[i]<a[j]){
            b[k++]=a[i++];

        }
        else{
            b[k++]=a[j++];
        }
    }
    while(i<=m){
        b[k++]=a[i++];
    }
    while(j<=r){
        b[k++]=a[j++];
    }
    for(int x=0;x<k;x++){
        a[l+x]=b[x];
    }



}
void mergesort(int a[],int l,int r){
    if(l<r){
        int mid=(l+r)/2;
        mergesort(a,l,mid);
        mergesort(a,mid+1,r);
        merge(a,l,mid,r);
    }
}
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    int m;
    scanf("%d",&m);
    int b[m];
    for(int i=0;i<m;i++){
        scanf("%d",&b[i]);
```

240701244

Scanned with OKEN Scanner

```
    }
    int c[m+n];
    int i;
    for(i=0;i<n;i++)
    {
        c[i]=a[i];
    }
    for(int j=0;j<m;j++){
        c[i+j]=b[j];
    }
    mergesort(c,0,(n+m)-1);
    printf("%d ",c[0]);
    for(int i=1;i<(m+n);i++){
        if(c[i]!=c[i-1]){
            printf("%d ",c[i]);
        }
    }
}
```

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

3. Problem Statement

Vishnu, a math enthusiast, is given a task to explore the magic of numbers. He has an array of positive integers, and his goal is to find the integer with the highest digit sum in the sorted array using the merge sort algorithm.

You have to assist Vishnu in implementing the merge sort algorithm.

Input Format

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

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

Output Format

The first line of output prints "The sorted array is: " followed by the sorted array, separated by a space.

Scanned with OKEN Scanner

The second line prints "The integer with the highest digit sum is: " followed by an integer representing the highest-digit sum.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
123 456 789 321 654

Output: The sorted array is: 123 321 456 654 789
The integer with the highest digit sum is: 789

Answer

```
// You are using GCC
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<i;j++)
        {
            if(a[i]<a[j])
            {
                int t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
    printf("The sorted array is: ");
    for(int i=0;i<n;i++)
    {
        printf("%d ",a[i]);
```

Scanned with OKEN Scanner

```
}
int b[n];
printf("\nThe integer with the highest digit sum is: ");
for(int i=0;i<n;i++)
{
    int k=a[i];
    int sum=0;
    while(k!=0)
    {
        int a;
        a=k%10;
        sum+=a;
        k=k/10;
    }
    b[i]=sum;
}
int max=0;
int index;
for(int i=0;i<n;i++)
{
    if (max<b[i]){
        max=b[i];
        index=i;
    }
}
printf("%d",a[index]);
}
```

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

4. Problem Statement

You're a coach managing a list of finishing times for athletes in a race. The times are stored in an array, and you need to sort this array in ascending order to determine the rankings.

You'll use the insertion sort algorithm to accomplish this.

Input Format

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

Scanned with OKEN Scanner

The second line contains n space-separated integers, each representing the finishing time of an athlete in seconds.

Output Format

The output prints the sorted finishing times of the athletes in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
75 89 65 90 70
Output: 65 70 75 89 90

Answer

```
// You are using GCC
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<i;j++)
            if(a[i]<a[j])
            {
                int t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
    }
    for(int i=0;i<n;i++)
    {
        printf("%d ",a[i]);
```

Scanned with OKEN Scanner

```
    }
}
```

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

5. Problem Statement

You are working on an optimization task for a sorting algorithm that uses insertion sort. Your goal is to determine the efficiency of the algorithm by counting the number of swaps needed to sort an array of integers.

Write a program that takes an array as input and calculates the number of swaps performed during the insertion sort process.

Example 1:

Input:

5

2 1 3 1 2

Output:

4

Explanation:

Step 1: [2, 1, 3, 1, 2] (No swaps)

Step 2: [1, 2, 3, 1, 2] (1 swap, element 1 shifts 1 place to the left)

Step 3: [1, 2, 3, 1, 2] (No swaps)

Step 4: [1, 1, 2, 3, 2] (2 swaps; element 1 shifts 2 places to the left)

Step 5: [1, 1, 2, 2, 3] (1 swap, element 2 shifts 1 place to the left)

Total number of swaps: 1 + 2 + 1 = 4

Example 2:

Input:

Scanned with OKEN Scanner

7

12 15 1 5 6 14 11

Output:

10

Explanation:

Step 1: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 2: [12, 15, 1, 5, 6, 14, 11] (1 swap, element 15 shifts 1 place to the left)

Step 3: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 4: [1, 12, 15, 5, 6, 14, 11] (2 swaps, element 1 shifts 2 places to the left)

Step 5: [1, 5, 12, 15, 6, 14, 11] (1 swap, element 5 shifts 1 place to the left)

Step 6: [1, 5, 6, 12, 15, 14, 11] (2 swaps, element 6 shifts 2 places to the left)

Step 7: [1, 5, 6, 12, 14, 15, 11] (1 swap, element 14 shifts 1 place to the left)

Step 8: [1, 5, 6, 11, 12, 14, 15] (3 swaps, element 11 shifts 3 places to the left)

Total number of swaps: 1 + 2 + 1 + 2 + 1 + 3 = 10

Input Format

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

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

Output Format

The output prints the number of swaps performed during the insertion sort process.

Refer to the sample output for the formatting specifications.

Sample Test Case

240701244

Scanned with OKEN Scanner

Input: 5
2 1 3 1 2
Output: 4

Answer

```
// You are using GCC
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    int c=0;
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++){
        for(int j=0;j<i;j++){
            if(a[i]<a[j]){
                int t=a[i];
                a[i]=a[j];
                a[j]=t;
                c+=1;
            }
        }
    }
    printf("%d",c);
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Marie, the teacher, wants her students to implement the ascending order of numbers while also exploring the concept of prime numbers.

Students need to write a program that sorts an array of integers using the merge sort algorithm while counting and returning the number of prime integers in the array. Help them to complete the program.

Input Format

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

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

Scanned with OKEN Scanner

Output Format

The first line of output prints the sorted array of integers in ascending order.

The second line prints the number of prime integers in the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
5 3 6 8 9 7 4
Output: Sorted array: 3 4 5 6 7 8 9
Number of prime integers: 3

Answer

```
// You are using GCC
#include<stdio.h>
#include<math.h>
int prime(int num){
   if(num<2) return 0;
   for(int i=2;i<=sqrt(num);i++){
      if(num%i==0){
         return 0;
      }
   }
   return 1;
}
int main()
{
   int n;
   scanf("%d",&n);
   int a[n];
   for(int i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   for(int i=0;i<n;i++){
      for(int j=0;j<i;j++){
         if(a[i]<a[j]){
```

240701244

Scanned with OKEN Scanner

```
            int t=a[i];
            a[i]=a[j];
            a[j]=t;
        }
    }
}
printf("Sorted array: ");
for(int i=0;i<n;i++){
    printf("%d ",a[i]);
}
printf("\n");
int c=0;
for(int i=0;i<n;i++){
    if(prime(a[i])){
        c++;
    }
  }
  printf("Number of prime integers: %d",c);
}
```

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

2. Problem Statement

Ravi is given an array of integers and is tasked with sorting it uniquely. He needs to sort the elements in such a way that the elements at odd positions are in descending order, and the elements at even positions are in ascending order.

Your task is to help Ravi create a program that uses insertion sort to sort the array as per the specified conditions and then print the sorted array. Position starts from 1.

Example

Input:

Size of the array = 10

Array elements = 25 36 96 58 74 14 35 15 75 95

240701244

Scanned with OKEN Scanner

Output:

Resultant array = 96 14 75 15 74 36 35 58 25 95

Explanation:

Initial Array: 25 36 96 58 74 14 35 15 75 95

Elements at odd positions (1, 3, 5, 7, 9): 25 96 74 35 75

Elements at odd positions sorted descending order: 96 75 74 35 25

Elements at even positions (2, 4, 6, 8, 10): 36 58 14 15 95

Elements at even positions sorted ascending order: 14 15 36 58 95

So, the final array is 96 14 75 15 74 36 35 58 25 95.

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 elements of the array.

Output Format

The output displays integers, representing the sorted array elements separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3 1 4 2

Output: 4 1 3 2

Answer

```
// You are using GCC
#include<stdio.h>
void sortd(int a[],int n){
    for(int i=0;i<n-1;i++){
```

240701244

Scanned with OKEN Scanner

```
        for(int j=i+1;j<n;j++){
            if(a[i]<a[j]){
                int t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
}
void sorta(int a[],int n){
    for(int i=0;i<n-1;i++){
        for(int j=i+1;j<n;j++){
            if(a[i]>a[j]){
                int t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
}
int main(){
    int n;
    scanf("%d",&n);
    int ar[n],o[n],e[n];
    int oc=0,ec=0;
    for(int i=0;i<n;i++){
        scanf("%d",&ar[i]);
        if((i+1)%2==1)
            o[oc++]=ar[i];
        else
            e[ec++]=ar[i];
    }
    sortd(o,oc);
    sorta(e,ec);
    int od=0,ev=0;
    for(int i=0;i<n;i++){
        if((i+1)%2==1)
        ar[i]=o[od++];
        else
        ar[i]=e[ev++];
    }
    for(int i=0;i<n;i++){
```

240701244

Scanned with OKEN Scanner

```
    printf("%d ",ar[i]);
  }
  return 0;
}
```

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

3. Problem Statement

Sheela wants to distribute cookies to her children, but each child will only be happy if the cookie size meets or exceeds their individual greed factor. She has a limited number of cookies and wants to make as many children happy as possible. Priya decides to sort both the greed factors and cookie sizes using QuickSort to efficiently match cookies with children. Your task is to help Sheela determine the maximum number of children that can be made happy.

Input Format

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

The second line contains n space-separated integers, where each integer represents the greed factor of a child.

The third line contains an integer m, representing the number of cookies.

The fourth line contains m space-separated integers, where each integer represents the size of a cookie.

Output Format

The output prints a single integer, representing the maximum number of children that can be made happy.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3

Scanned with OKEN Scanner

1 2 3
2
1 1
Output: The child with greed factor: 1

Answer

```
// You are using GCC
#include<stdio.h>
void swap(int*a,int*b){
    int t=*a;
    *a=*b;
    *b=t;
}

int partition(int arr[],int low,int high){
    int pivot=arr[high];
    int i=low- 1;
    for(int j=low;j<=high- 1;j++){
        if(arr[j]<pivot){
            i++;
            swap(&arr[i],&arr[j]);
        }
    }
    swap(&arr[i+ 1],&arr[high]);
    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);
    }
}
int main(){
    int n,m;
    scanf("%d",&n);
    int g[n];
    for(int i=0;i<n;i++){
        scanf("%d",&g[i]);
    }
    scanf("%d",&m);
    int s[m];
```

Scanned with OKEN Scanner

```
for(int i=0;i<m;i++){
    scanf("%d",&s[i]);
}
quicksort(g, 0, n- 1);
quicksort(s, 0, m- 1);
int i=0,j=0,c=0;
while(i<n&&j<m){
   if(s[j]>=g[i]){

      c++;
      i++;
   }
   j++;

}
    printf("The child with greed factor: %d \n",c);

  return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 2
Total Mark : 20
Marks Obtained : 20

Section 1 : MCQ

1. 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

1

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

2. 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***

Scanned with OKEN Scanner

3. Which situation causes clustering in linear probing?

Answer

All the mentioned options

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

4. What is the primary disadvantage of linear probing?

Answer

Clustering

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

5. 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***

6. Which of these hashing methods may result in more uniform distribution with small keys?

Answer

Mid-Square

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

7. Which C statement is correct for finding the next index in linear probing?

Answer

index = (index + 1) % size;

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

Scanned with OKEN Scanner

8. In division method, if key = 125 and m = 13, what is the hash index?

Answer

8

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

9. 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***

10. 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***

11. What does a deleted slot in linear probing typically contain?

Answer

A special "deleted" marker

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

12. 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***

Scanned with OKEN Scanner

13. 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***

14. What happens if we do not use modular arithmetic in linear probing?

Answer

Index goes out of bounds

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

15. What is the initial position for a key k in a linear probing hash table?

Answer

k % table_size

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

16. 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***

17. In the division method of hashing, the hash function is typically written as:

Answer

h(k) = k % m

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

Scanned with OKEN Scanner

18. Which data structure is primarily used in linear probing?

Answer

Array

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

19. Which of the following values of 'm' is recommended for the division method in hashing?

Answer

A prime number

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

20. What would be the result of folding 123456 into three parts and summing: (12 + 34 + 56)?

Answer

102

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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

// You are using GCC
void initializeTable(int table[], int size) {
    //Type your code here
    for(int i=0;i<size;i++){
        table[i]=-1;
    }
}

int linearProbe(int table[], int size, int num) {
    //Type your code here
    int index=num%size;
    int start=index;
    while(table[index]!=-1){
        index=(index+1)%size;
        if(index==start)
```

Scanned with OKEN Scanner

```
        return -1;
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    //Type your code here
    for(int i=0;i<n;i++){
        int index=linearProbe(table,size,arr[i]);
        if(index!=-1){
            table[index]=arr[i];
        }
    }
}

void printTable(int table[], int size) {
    //Type your code here
    for(int i=0;i<size;i++){
        printf("%d",table[i]);
        if(i<size-1)
            printf(" ");
    }
}

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***

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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.

Scanned with OKEN Scanner

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

// You are using GCC
void initializeTable(int table[], int size) {
    //Type your code here
    for(int i=0;i<size;i++){
        table[i]=-1;
    }
}

int linearProbe(int table[], int size, int num) {
    //Type your code here
    int index=num%size;
```

Scanned with OKEN Scanner

```
    int start=index;
    while(table[index]!=-1){
        index=(index+1)%size;
        if(index==start)
            return -1;
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    //Type your code here
    for(int i=0;i<n;i++){
        int index=linearProbe(table,size,arr[i]);
        if(index!=-1){
            table[index]=arr[i];
        }
    }
}

int searchInHashTable(int table[], int size, int num) {
    //Type your code here
    int index=num%size;
    int start=index;
    while(table[index]!=-1){
        if(table[index]==num)
        return 1;
        index=(index+1)%size;
        if(index==start)
        break;
    }
    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);
```

240701244

Scanned with OKEN Scanner

```
    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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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).

Scanned with OKEN Scanner

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
void insertKeyValuePair(Dictionary *dict, const char *key, const char *value) {
   //Type your code here
   if(dict->size==dict->capacity){
      dict->capacity*=2;
      dict->pairs=(KeyValuePair*)realloc(dict->pairs,dict-
>capacity*sizeof(KeyValuePair));
```

Scanned with OKEN Scanner

```
    }
    strcpy(dict->pairs[dict->size].key,key);
    strcpy(dict->pairs[dict->size].value,value);
    dict->size++;
}
void removeKeyValuePair(Dictionary *dict, const char *key) {
    //Type your code here
    int found=0;
    for(int i=0;i<dict->size;i++){
        if(strcmp(dict->pairs[i].key,key)==0){
            found=1;
        }
        if(found&&i<dict->size-1){
            dict->pairs[i]=dict->pairs[i+1];
        }
    }
    if(found)
        dict->size--;
}
int doesKeyExist(Dictionary *dict, const char *key) {
    //Type your code here
    for(int i=0;i<dict->size;i++){
        if(strcmp(dict->pairs[i].key,key)==0)
            return 1;
    }
    return 0;
}
void printDictionary(Dictionary *dict) {
    //Type your code here
    for(int i=0;i<dict->size;i++){
        printf("Key: %s; Value: %s\n",dict->pairs[i].key,dict->pairs[i].value);
    }
}
```

Status : Correct

Marks : 10/10

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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.

Scanned with OKEN Scanner

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
int keyExists(KeyValuePair* dictionary, int size, const char* key) {
    //Type your code here
    for(int i=0;i<size;i++){
        if(strcmp(dictionary[i].key,key)==0){
            return 1;
        }
    }
    return 0;
}
```

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

Scanned with OKEN Scanner

Rajalakshmi Engineering College

Name: Kavitha R
Email: 240701244@rajalakshmi.edu.in
Roll no: 240701244
Phone: 8610034812
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

Scanned with OKEN Scanner

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

240701244

Scanned with OKEN Scanner

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

// You are using GCC
unsigned int hash(int key, int tableSize) {
    //Type your code here
    int squared=key^key;
    return squared%tableSize;
}

int getOddOccurrence(int arr[], int size) {
    //Type your code here
    int table[MAX_SIZE]={0};
    int tableSize=MAX_SIZE;
    for(int i=0;i<size;i++){
        int h=hash(arr[i],tableSize);
        table[h]^=arr[i];
    }
    for(int i=0;i<tableSize;i++){
        if(table[i]!=0){
            return table[i];
        }
    }
```

Scanned with OKEN Scanner

```
    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

Scanned with OKEN Scanner