Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

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

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

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

Input Format

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

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

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

Output Format

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

```
#include <stdio.h>

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

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

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

    calculateRowSum(matrix, r, c);
```

```
    return 0;
}
```

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

2. Problem Statement

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

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

Input Format

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

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

The subsequent lines contain integers representing the menu choices:

Choice 1: Find and display the smallest number.

Choice 2: Find and display the largest number.

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

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

Choice 5: Exit the program.

Output Format

For each valid menu choice, print the corresponding result:

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

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

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

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

For choice 5, print "Exiting the program".

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

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
10 20 30
1
5
Output: The smallest number is: 10
Exiting the program

Answer

```
#include<stdio.h>
int small(int a[],int n){
   int s = a[0];
   for(int i=0;i<n;i++){
      if(a[i]<s){
         s = a[i];
      }
   }
   return s;

}
int large(int a[],int n){
```

```
    int l=a[0];
    for(int i=0;i<n;i++){
        if(a[i]>l){
            l = a[i];
        }
    }
    return l;
}
int sum(int a[],int n){
    int s=0;
    for(int i=0;i<n;i++){
        s +=a[i];
    }
    return s;
}
double avg(int a[],int n){
    int s = sum(a,n);
    return (double)s/n;
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    int c;
    while(1){
        scanf("%d",&c);
        switch(c){
            case 1:
            printf("The smallest number is: %d\n",small(a,n));
            break;
            case 2:
            printf("The largest number is: %d\n",large(a,n));
            break;
            case 3:
            printf("The sum of the numbers is: %d\n",sum(a,n));
            break;
            case 4:
            printf("The average of the numbers is: %.2lf\n",avg(a,n));
            break;
```

```
            case 5:
            printf("Exiting the program\n");
            return 0;
            default:
            printf("Invalid choice! Please enter a valid option (1-5).\n");

        }
    }
    return 0;
}
```

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

3. Problem Statement

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

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

Input Format

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

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

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

Output Format

The output displays one of the following results:

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

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

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Answer

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

         if((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

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

Help Saurabh automate this task efficiently.

Input Format

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

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

Output Format

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

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

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

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

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

Where max_salary is the highest salary among all employees.

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

Where min_salary is the lowest salary among all employees.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
4000
3500
6000
2500
4500

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

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

Answer

```
#include<stdio.h>
int main(){
    int 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++){
        printf("Employee %d: %d\n",i+1,a[i]);
```

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

        int max = a[0];
    for(int i=0;i<n;i++){

        if(a[i]>max){
            max=a[i];
        }
    }
    int min = a[0];
    for(int i=0;i<n;i++){

        if(a[i]<min){
            min = a[i];
        }
    }
    int sum =0;
    for(int i=0;i<n;i++){
        sum += a[i];
    }
    double avg = (double)sum / n;
    printf("Average salary: %.2lf\n",avg);
    printf("Highest salary: %d\n",max);
    printf("Lowest salary: %d\n",min);
}
```

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

5. Problem Statement

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

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

Input Format

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

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

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Output: 0 20 30
0 0 60
0 0 0

Answer

```
#include <stdio.h>

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

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

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
```

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

    setZeros(arr1, n);

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

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Vijay G
Email: 241901124@rajalakshmi.edu.in
Roll no: 241901124
Phone: 7548817843
Branch: REC
Department: I CSE (CS) AC
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Pointers

Attempt : 2
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. 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:

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

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
    int a;
    scanf("%d", &a);
    double* b = (double*)malloc(a*sizeof(double));
    for (int i = 0; i < a; i++) {
```

```
        scanf("%lf", &b[i]);
    }
    for (int i = 0; i < a; i++) {
        if (i == 0) {
            printf("%.2f ", b[i]);
        }else {
            printf("%.2f ", fabsf(b[i] - b[i-1]));
        }
    }
    printf("\n");
    float lastD = 0;
    for (int i = 0; i < a; i++) {
        if (i == 0 || i == (a-1)) {
            printf("%.2f ", b[i]);
            lastD = b[i];
        }else {
            lastD = ((lastD + b[i+1])/2.0f);
            printf("%.2f ", lastD);
        }
    }
}
```

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

2. 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>

void calculatePermutations(int n, int r) {
    unsigned long long product = 1;
    int temp = n;
    int *current = &temp;
    for (int i = 0; i < r; ++i) {
        product *= *current;
        (*current)--;
    }
    printf("%llu", product);
}
int main() {
    int n, r;
    scanf("%d", &n);
    scanf("%d", &r);
    calculatePermutations(n, r);
    return 0;
}
```

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

3. 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

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>

struct Person {
   int *age;
};

int main() {
   int age_input;
   scanf("%d", &age_input);
   struct Person p;
   p.age = &age_input;
   if (*p.age < 18) {
   printf("Invalid\n");
   }
   else if (*p.age >= 65) {
   printf("Already retired!\n");
   } else {
      int remaining_years = 65 - *p.age;
      int retirement_year = 2023 + remaining_years;
      int remaining_days = remaining_years * 365;
      printf("Retirement Year: %d\n", retirement_year);
      printf("Remaining Years: %d\n", remaining_years);
      printf("Remaining Days: %d\n", remaining_days);
   }
   return 0;
}
```

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.

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>
#include <string.h>

struct Employee {
    char name[51];
```

```
    double salary;
    int hours;
};

int main() {
    struct Employee emp;
    struct Employee *ptr = &emp;
    // Read employee name
    fgets(ptr->name, 51, stdin);
    ptr->name[strcspn(ptr->name, "\n")] = '\0'; // Remove newline character
    // Read salary and hours
    scanf("%lf", &ptr->salary);
    scanf("%d", &ptr->hours);
    // Calculate salary increase based on hours
    double increase = 0.0;
    if (ptr->hours >= 12) {
        increase = 150.00;
    } else if (ptr->hours >= 10) {
        increase = 100.00;
    } else if (ptr->hours >= 8) {
        increase = 50.00;
    }

    // Update final salary
    ptr->salary += increase;

    // Calculate total salary for 30 days
    double total = ptr->salary * 30;

    // Output the results with two decimal places
    printf("Final Salary: Rs. %.2lf\n", ptr->salary);
    printf("Total Salary: Rs. %.2lf\n", total);
    return 0;
}
```

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

5. 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."

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

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

int main() {
    int a;
    scanf("%d", &a);
    float* b = (float*)malloc(a*sizeof(float));
    int aP = 1, hP = 0;
    for (int i = 0; i < a; i++) {
        scanf("%f", &b[i]);
```

```
    }

    for (int i = 0; i < a; i++) {
        if (b[i] > 0) {
                    hP = 1;
        }else {
                    aP = 0;
        }
    }
    if (aP) {
            printf("All elements are positive.");
    }else if (hP) {
             printf("At least one element is positive.");
    }else {
             printf("No positive elements in the array.");
    }
}
```

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