NAME: R.KARTHICK RAJA
ROLL NO.: 240801147

WEEK 0-ARRAYS AND FUNCTIONS

QUESTION 1:

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.

Code constraints :

In this scenario, the test cases fall under the following constraints:
1 ≤ n,m ≤ 5
1 ≤ Class Sessions ≤ 100

Sample test cases :

Input 1 :

```
2
2
1 2
3 100
```

Output 1 :

```
100 is even
```

Input 2 :

```
5
```

```
3
10 2 3
4 5 6
7 8 9
10 11 12
1 2 3
```

Output 2 :

```
3 is odd
```

SOURCE CODE:

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

QUESTION 2:

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.

Code constraints :

1 ≤ row, column ≤ 20

Sample test cases :

Input 1 :

```
3
3
1 2 3
4 5 6
7 8 9
```

Output 1 :

```
6 15 24
```

SOURCE CODE:

```
void calculateRowSum(int matrix[20][20], int rows, int cols) {
    int s;
    for(int i=0;i<rows;i++){
        s=0;
        for(int j=0;j<cols;j++){
            s+=matrix[i][j];
        }
        printf("%d ",s);
    }
}
```

QUESTION 3:

Problem Statement

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

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

Input format :

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

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

Output format :
The output displays the matrix containing M space-separated elements in M lines where the lower triangular elements are replaced with zero.
Refer to the sample output for formatting specifications.
Code constraints :
1 ≤ M ≤ 10
Sample test cases :
Input 1 :

```
3
10 20 30
40 50 60
70 80 90
```

Output 1 :

```
0 20 30
0 0 60
0 0 0
```

SOURCE CODE:

```
void setZeros(int arr[10][10], int n) {
    for(int i=0;i<n;i++){
        for(int j=0;j<=i;j++){
            arr[i][j]=0;
        }
    }
}
```

QUESTION 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.
Code constraints :
The given test cases fall under the following specifications:
1 ≤ n ≤ 100
100 ≤ salary ≤ 1000000
Sample test cases :
Input 1 :

```
5
4000
3500
6000
2500
4500
```

Output 1 :

```
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
```

Input 2 :

```
3
1000
2000
3000
```

Output 2 :

```
Employee 1: 1000
Employee 2: 2000
Employee 3: 3000

Average Salary: 2000.00
Highest Salary: 3000
Lowest Salary: 1000
```

SOURCE CODE:

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

QUESTION 5:

Problem Statement

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

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

Input format :

The first line contains an integer n, representing the number of elements in the array.
The second line contains n space-separated integers representing the array elements.
The subsequent lines contain integers representing the menu choices:
Choice 1: Find and display the smallest number.
Choice 2: Find and display the largest number.

Choice 3: Calculate and display the sum of all numbers.
Choice 4: Calculate and display the average of all numbers as double.
Choice 5: Exit the program.

Output format :

For each valid menu choice, print the corresponding result:
For choice 1, print "The smallest number is: X", where X is the smallest number in the array.
For choice 2, print "The largest number is: X", where X is the largest number in the array.
For choice 3, print "The sum of the numbers is: X", where X is the sum of all numbers in the array.
For choice 4, print "The average of the numbers is: X. XX", where X.XX is the double value representing an average of all numbers in the array, rounded to two decimal places.
For choice 5, print "Exiting the program".
If an invalid choice is made, print "Invalid choice! Please enter a valid option (1-5)."
Refer to the sample output for the formatting specifications.

Code constraints :

1 ≤ n ≤ 10
−104 ≤ array elements ≤ 104
The menu choices are integers 1,2,3,4 or 5.

Sample test cases :

Input 1 :

```
3
10 20 30
1
5
```

Output 1 :

```
The smallest number is: 10
Exiting the program
```

Input 2 :

```
4
1 2 3 4
2
5
```

Output 2 :

```
The largest number is: 4
Exiting the program
```

Input 3 :

```
5
5 5 5 5 5
3
5
```

Output 3 :

```
The sum of the numbers is: 25
Exiting the program
```

Input 4 :

```
2
```

```
8 12
4
5
```

Output 4 :

```
The average of the numbers is: 10.00
Exiting the program
```

Input 5 :

```
6
3 6 9 12 15 18
1
2
3
4
5
```

Output 5 :

```
The smallest number is: 3
The largest number is: 18
The sum of the numbers is: 63
The average of the numbers is: 10.50
Exiting the program
```

Input 6 :

```
5
100 200 300 400 500
6
5
```

Output 6 :

```
Invalid choice! Please enter a valid option (1-5).
Exiting the program
```

SOURCE CODE:

```
#include <stdio.h>
void smallest(int a[],int n){
    int s=a[0];
    for(int i=1;i<n;i++){
        if(a[i]<s)
        s=a[i];
    }
    printf("The smallest number is: %d\n",s);
}
void largest(int a[],int n){
    int l=a[0];
    for(int i=1;i<n;i++){
        if(a[i]>l)
        l=a[i];
    }
    printf("The largest number is: %d\n",l);
}
void sum(int a[],int n){
    int s=0;
    for(int i=0;i<n;i++){
        s+=a[i];
    }
    printf("The sum of the numbers is: %d\n",s);
}
void average(int a[],int n){
    float r=0;
    for(int i=0;i<n;i++){
        r+=a[i];
    }
    printf("The average of the numbers is: %.2f\n",r/n);
}
int main(){
    int n,i,m;
    scanf("%d",&n);
    int a[n];
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
    do{
        scanf("%d",&m);
        switch(m){
            case 1:
            smallest(a,n);
            break;
            case 2:
            largest(a,n);
            break;
            case 3:
            sum(a,n);
            break;
            case 4:
            average(a,n);
            break;
            case 5:
            printf("Exiting the program");
            break;
            default:
            printf("Invalid choice! Please enter a valid option (1-5).");
            break;
        }
    }while(m!=5);
    return 0;
}
```