Advanced C Lab assignment 2 Ex3

Name: Sankalp Mukim

Registration Number: 20BDS0128

Course Code: CSE2010

Slot: L27+L28

Table of Contents

[Question 1 1](#_Toc97880946)

[Code 1](#_Toc97880947)

[Output 4](#_Toc97880948)

[Question 2 4](#_Toc97880949)

[Code 4](#_Toc97880950)

[Output 5](#_Toc97880951)

[Question 3 5](#_Toc97880952)

[Code 5](#_Toc97880953)

[Output 8](#_Toc97880954)

[Question 4 8](#_Toc97880955)

[Code 8](#_Toc97880956)

[Output 11](#_Toc97880957)

# Question 1

Write a function to convert any given year into its roman equivalent. Use these roman equivalents for decimal numbers: 1-I, 5-V, 10-X,50-L, 100-C, 500-D, 1000-M Example: Roman equivalent of 1988 is MDCCCCLXXXVIII

## Code

**//** Write a recursive function to convert any given year into its roman equivalent. Use these roman equivalents for decimal values:

**//** 1 = I, 5 = V, 10 = X, 50 = L, 100 = C, 500 = D, 1000 = M.

#include **<**stdio.h**>**

***void*** convertToRoman(***int*** year, ***char*** \*roman, ***int*** length)

{

    if (year > 1000)

    {

        roman[length] = **'**M**'**;

        length++;

        year -= 1000;

        convertToRoman(year, roman, length);

    }

    else if (year > 500)

    {

        roman[length] = **'**D**'**;

        length++;

        year -= 500;

        convertToRoman(year, roman, length);

    }

    else if (year > 100)

    {

        roman[length] = **'**C**'**;

        length++;

        year -= 100;

        convertToRoman(year, roman, length);

    }

    else if (year > 50)

    {

        roman[length] = **'**L**'**;

        length++;

        year -= 50;

        convertToRoman(year, roman, length);

    }

    else if (year > 10)

    {

        roman[length] = **'**X**'**;

        length++;

        year -= 10;

        convertToRoman(year, roman, length);

    }

    else if (year > 5)

    {

        roman[length] = **'**V**'**;

        length++;

        year -= 5;

        convertToRoman(year, roman, length);

    }

    else if (year > 1)

    {

        roman[length] = **'**I**'**;

        length++;

        year -= 1;

        convertToRoman(year, roman, length);

    }

    else if (year == 1)

    {

        roman[length] = **'**I**'**;

        length++;

    }

    else if (year == 5)

    {

        roman[length] = **'**V**'**;

        length++;

    }

    else if (year == 10)

    {

        roman[length] = **'**X**'**;

        length++;

    }

    else if (year == 50)

    {

        roman[length] = **'**L**'**;

        length++;

    }

    else if (year == 100)

    {

        roman[length] = **'**C**'**;

        length++;

    }

    else if (year == 500)

    {

        roman[length] = **'**D**'**;

        length++;

    }

    else if (year == 1000)

    {

        roman[length] = **'**M**'**;

        length++;

    }

    else if (year == 0)

    {

        roman[length] = **'**\0**'**;

    }

}

***int*** main()

{

***int*** year;

    printf(**"**Enter a year: **"**);

    scanf(**"**%d**"**, &year);

***char*** roman[100];

    convertToRoman(year, roman, 0);

    printf(**"**%s\n**"**, roman);

    return 0;

}

## Output

Text

Description automatically generated

# Question 2

A positive integer is entered through the keyboard. Write a function to obtain the prime factors of this number. Example: prime factors of 24 are 2,2,2 and 3

## Code

**//** A positive integer is entered through the keyboard. Write a function to obtain the prime factors of this number.

**//** Example: prime factors of 24 are 2, 2, 2 and 3.

#include **<**stdio.h**>**

***void*** primeFactors(***int*** num)

{

***int*** i;

    for (i = 2; i <= num; i++)

    {

        while (num % i == 0)

        {

            printf(**"**%d **"**, i);

            num = num / i;

        }

    }

}

***int*** main()

{

***int*** num;

    printf(**"**Enter a number: **"**);

    scanf(**"**%d**"**, &num);

    primeFactors(num);

    return 0;

}

## Output

Text

Description automatically generated

# Question 3

Write a program and the following functions to compute the average rainfall for the year. Use an array to store pointers to the first day of each month and another array to store the number of days in each month.

a. A function to input the average rainfall data of each day of the year into a one-dimensional array.

b. A function to compute the average rainfall for the year or any month.

c. A function to output the average rainfall for each month and the yearly average rainfall.

## Code

**//** Write a program and the following functions to compute the average rainfall for the year. Use an array to store pointers to the first day of each month and another array to store the number of days in each month.

**//** a. A function to input the average rainfall data of each day of the year into a one dimensional

**//** array.

**//** b. A function to compute the average rainfall for the year or any month.

**//** c. A function to output the average rainfall for each month and the yearly average rainfall.

#include **<**stdio.h**>**

#include **<**stdlib.h**>**

***double*** \*\*inputRainfall();

***double*** averageRainfall(***double*** \*\*rainfall);

***double*** averageRainfallByMonth(***double*** \*\*rainfall, ***int*** month);

***void*** outputRainfall(***double*** \*\*rainfall);

**//** array to store the number of days in each month

***int*** days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

***int*** main()

{

***double*** \*\*rainfall = inputRainfall();

    outputRainfall(rainfall);

    return 0;

}

**//** input all rainfall month by month

***double*** \*\*inputRainfall()

{

**//** rainfall is the array to store pointers to the first day of each month

***double*** \*\*rainfall = (***double*** \*\*)malloc(12 \* sizeof(***double*** \*));

***int*** i;

    for (i = 0; i < 12; i++)

    {

        rainfall[i] = (***double*** \*)malloc(days[i] \* sizeof(***double***));

    }

    for (i = 0; i < 12; i++)

    {

***int*** j;

        printf(**"**Enter rainfall for month %d: **"**, i + 1);

        for (j = 0; j < days[i]; j++)

        {

            scanf(**"**%lf**"**, &rainfall[i][j]);

        }

    }

    return rainfall;

}

***void*** outputRainfall(***double*** \*\*rainfall)

{

***double*** average = averageRainfall(rainfall);

    printf(**"**The average rainfall for the year is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 1);

    printf(**"**The average rainfall for January is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 2);

    printf(**"**The average rainfall for February is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 3);

    printf(**"**The average rainfall for March is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 4);

    printf(**"**The average rainfall for April is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 5);

    printf(**"**The average rainfall for May is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 6);

    printf(**"**The average rainfall for June is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 7);

    printf(**"**The average rainfall for July is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 8);

    printf(**"**The average rainfall for August is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 9);

    printf(**"**The average rainfall for September is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 10);

    printf(**"**The average rainfall for October is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 11);

    printf(**"**The average rainfall for November is %.2f\n**"**, average);

    average = averageRainfallByMonth(rainfall, 12);

    printf(**"**The average rainfall for December is %.2f\n**"**, average);

}

***double*** averageRainfall(***double*** \*\*rainfall)

{

***double*** average = 0;

***int*** i, j;

    for (i = 0; i < 12; i++)

    {

        for (j = 0; j < days[i]; j++)

        {

            average += rainfall[i][j];

        }

    }

    average /= 365;

    return average;

}

***double*** averageRainfallByMonth(***double*** \*\*rainfall, ***int*** month)

{

***double*** average = 0;

***int*** i;

    for (i = 0; i < days[month - 1]; i++)

    {

        average += rainfall[month - 1][i];

    }

    average /= days[month - 1];

    return average;

}

## Output

Text

Description automatically generated

# Question 4

Write a program and the following functions to compute the average value for the following data values stored in a two-dimensional array.

a. A function to input the data into a two-dimensional array.

b. A function to compute the row averages and store them in a one-dimensional array.

c. A function to compute the column averages and store them in a one-dimensional array.

d. A function to compute the average of all the values in the array.

e. A function to output the array, row averages, column averages, and the overall average.

## Code

**//** Write a program and the following functions to compute values for the following data values stored in a two dimensional array.

#include **<**stdio.h**>**

#include **<**stdlib.h**>**

**//** A function to input the data into a two dimensional array.

***double*** \*\*input2DData(***int*** rows, ***int*** columns);

**//** A function to compute the row averages and store them in a one dimensional array.

***void*** computeRowAverages(***double*** \*rowAverages, ***int*** columns, ***double*** \*\*data);

**//** A function to compute the column averages and store them in a one dimensional array.

***void*** computeColumnAverages(***double*** \*columnAverages, ***int*** rows, ***double*** \*\*data);

**//** A function to compute the average of all the values in the array.

***double*** fullAverage(***double*** \*\*data, ***int*** rows, ***int*** columns);

**//** A function to output the array, row averages, column averages, and the overall average.

***void*** outputData(***double*** \*\*data, ***int*** rows, ***int*** columns);

***int*** main()

{

***int*** rows, columns;

    printf(**"**Enter the number of rows:**"**);

    scanf(**"**%d**"**, &rows);

    printf(**"**Enter the number of columns:**"**);

    scanf(**"**%d**"**, &columns);

***double*** \*\*data = input2DData(rows, columns);

    outputData(data, rows, columns);

}

***double*** \*\*input2DData(***int*** rows, ***int*** columns)

{

**//** allocate space

***double*** \*\*data = (***double*** \*\*)malloc(rows \* sizeof(***double*** \*));

    printf(**"**Total rows to enter data for: %d\n**"**, rows);

**//** input data

    for (***size\_t*** i = 0; i < rows; i++)

    {

        printf(**"**Enter data for row %ld: **"**, (i + 1));

        data[i] = (***double*** \*)malloc(sizeof(***double***));

        for (***size\_t*** j = 0; j < columns; j++)

        {

            scanf(**"**%lf**"**, &data[i][j]);

        }

    }

    return data;

}

***void*** computeRowAverages(***double*** \*rowAverages, ***int*** columns, ***double*** \*\*data)

{

    for (***size\_t*** i = 0; i < columns; i++)

    {

***double*** sum = 0;

        for (***size\_t*** j = 0; j < columns; j++)

        {

            sum += data[i][j];

        }

        rowAverages[i] = sum / columns;

    }

}

***void*** computeColumnAverages(***double*** \*columnAverages, ***int*** rows, ***double*** \*\*data)

{

    for (***size\_t*** i = 0; i < rows; i++)

    {

***double*** sum = 0;

        for (***size\_t*** j = 0; j < rows; j++)

        {

            sum += data[j][i];

        }

        columnAverages[i] = sum / rows;

    }

}

***double*** fullAverage(***double*** \*\*data, ***int*** rows, ***int*** columns)

{

***double*** sum = 0;

    for (***size\_t*** i = 0; i < rows; i++)

    {

        for (***size\_t*** j = 0; j < columns; j++)

        {

            sum += data[i][j];

        }

    }

    return sum / (rows \* columns);

}

***void*** outputData(***double*** \*\*data, ***int*** rows, ***int*** columns)

{

***double*** \*rowAverages = (***double*** \*)malloc(sizeof(***double***) \* columns);

***double*** \*columnAverages = (***double*** \*)malloc(sizeof(***double***) \* rows);

    computeRowAverages(rowAverages, columns, data);

    computeColumnAverages(columnAverages, rows, data);

    printf(**"**\nRow averages:\n**"**);

    for (***size\_t*** i = 0; i < columns; i++)

    {

        printf(**"**%f\n**"**, rowAverages[i]);

    }

    printf(**"**\nColumn averages:\n**"**);

    for (***size\_t*** i = 0; i < rows; i++)

    {

        printf(**"**%f\n**"**, columnAverages[i]);

    }

    printf(**"**\nOverall average: %f\n**"**, fullAverage(data, rows, columns));

}

## Output

Graphical user interface, text, website

Description automatically generated