Computer Practicum 1: C Homework
Fall Semesters AY 2020-2021

Instructions: There are many problems in this document. Solve as many problems as you can. Use a separate file for each problem. The problems are labeled 00 to 16. When naming your files, use the format surname_firstname_problemXX.c (example: `Deja_Jordan_problem00.c`). You have 24 hours to complete this task. When done, compile all programs in one zip/rar archive. The name of the archive should use the format surname_firstname_CHW.zip/rar. Submit the compressed file in the e-classroom. The e-classroom assignment link will be open for 25 hours. Absolutely no extensions or special submissions will be allowed.

On Grading: The programs will be graded based on the following criteria:

- Originality of Solution: be very creative in your solutions. If there are already existing solutions, we challenge you to use a more diverse approach.
- Code Documentation: We appreciate comments that accurately describe what the program does, what special variables or functions do, what special equations do and many others.
- Variable/Constant Name: Hungarian notation and descriptive identifier names will be appreciated. We shall also look at cases where similar data types are arranged by type and if some important identifiers have an initialized value.
- On Problem Difficulty: There will be very difficult and very easy problems in this document. For the difficult problems, if you are unable to solve them, partial or commented solutions will be accepted and be given extra points. For example, if you are unable to finish it but have an idea how to solve it, please submit your .c code with comments on how you intend to do the solution. I would like to see how your mind works when dealing with such problems.

The goal is to practice and enhance your skills using a variety of problems and approaches. In this homework, you will be exposed to various problems involving characters, mathematical equations and even some “real-life” scenarios. Take the time to solve and enhance your skills. Goodluck!

Proceed to the next page for the set of problems.

Problem 00.c

Create a C Program that will do the following below:

The program should allow the user to input a value for variables a, b, c and be able to determine if the sides/angles represent a Right Triangle, Acute Triangle or an Obtuse Triangle.

Note the following facts:

1. A triangle has 3 sides a, b, c and 3 angles A, B, C, where angle A is opposite side a, angle B is opposite side b, and angle C is opposite side c.
2. A triangle is a right triangle if one angle measures 90 degrees. A triangle is an obtuse triangle if one angle measures more than 90 degrees. A triangle is an acute triangle if all angles measure less than 90 degrees.
3. Cosine Law states that:
    a. $c^2 = a^2 + b^2 - 2ab(\cos C)$
    b. $b^2 = a^2 + c^2 - 2ac(\cos B)$
    c. $a^2 = b^2 + c^2 - 2bc(\cos A)$
4. Sine Law states that: $a/\sin A = b/\sin B = c/\sin C$
5. 180 degrees is equal to π radians.

See the sample input/output below:

```
Input side a: 3
Input side b: 4
Input angle C: 90
Right Triangle
```

```
Input side a: 4
Input side b: 4
Input angle C: 50
Acute Triangle
```

```
Input side a: 10
Input side b: 10
Input angle C: 100
Obtuse Triangle
```

Problem 01.c

Create a C Program that will perform the following operations below:

Animal insurance: A program that prints the insurance fee to pay for a pet according to the following rules:

A dog that has been neutered costs $50.
A dog that has not been neutered costs $80.
A cat that has been neutered costs $40.
A cat that has not been neutered costs $60.
A bird or reptile costs nothing.
Any other animal generates an error message.

The program should prompt the user for the appropriate information, using a code to determine the kind of animal i.e. D or d represents a dog, C or c represents a cat, B or b represents a bird, R or r represents a reptile, and anything else represents some other kind of animal .

You may add other features that will make your code more elegant and your program more interactive.

Problem 02.c

Create a C Program that will perform the following operations below:

The program should allow the user to enter a value of *n* and writes this picture:

```
N=1
*
N=2
**
 *
N=3
***
 **
  *
```

You may add other features that will make your code more elegant and your program more interactive.

Problem 03.c

Write a program that asks the user to type a positive integer. When the user types a negative value the program writes ERROR and asks for another value. When the user types 0, that means that the last value has been typed and the program must write the average of the positive integers. If the number of typed values is zero the program writes 'NO AVERAGE'.

Problem 04.c

Write a program that asks the user to type an integer N and that writes the number of prime numbers lesser or equal to N.

Problem 05.c

Write a C program that will perform the following functions below:

Allow the user to enter a number. This number will determine the maximum number of asterisks in a row. Print a left-based triangle that will have the user's input number as the maximum number. For example, if the user inputs the number 5, the output should be as displayed below:

```
*
**
***
****
*****
****
***
**
*
```

Problem 06.c

Write a program that asks the user to type 10 integers and returns the smallest integer in the list.

Problem 07.c

Request the user to enter one or two-digit integers, then print that input but three times, until the user enters -999. If the user enters '1' then it should print "111". If the user enters '8' then it should print '888'. The program will never stop asking for an input until the user enters 999.

Problem 08.c

Multiplication Table: A program that will ask the user to input an integer between 1 to 10 only. If the user inputs “2”, it will display the multiplication table values from 1 from 1 to 10 and the values of 2 from 1 to 10. If the user inputs “4”, it will display like as shown below:

x	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16
5	5	10	15	20
6	6	12	18	24
7	7	14	21	28
8	8	16	24	32
9	9	18	27	36
10	10	20	30	40

The headers on the top are multiplied to the first column, and the values in the middle are its respective product values.

Problem 09.c

Calendar Generator: A program that will show the specific month. It will ask the user which day of the week it will start, the date and how many days does it have. Refer to the sample below:

sample execution:

```
Calendar of a specific month will be shown.
Which day of the week does the month start?
Input figure among following figures.
0: Sun, 1: Mon, 2: Tue, 3: Wed, 4: Thu, 5: Fri, 6: Sat
5
How many days does the month have?
Input figure
28

Su    Mo    Tu    We    Th    Fr    Sa
                                1     2
3     4     5     6     7     8     9
10    11    12    13    14    15    16
17    18    19    20    21    22    23
24    25    27    28    $
```

Problem 10.c

Simulate an elevator-controller program. The Elevator should be able to travel between five floors namely Ground, 1st, 2nd, 3rd and 4th floor. The elevator can either travel going up or going down. The elevator can only contain one passenger (which is the current user of the program). Initially, the elevator begins and loads at the ground floor. If the user wants to travel to the 4th floor, the elevator shall go to the said floor. It is the option of the user if they want to leave the elevator or not.

You may apply necessary visuals when needed. Realistic rules should apply: If the user leaves at the 3rd floor, when it rides the elevator again, it should begin travelling from the 3rd floor. Your program should have validation controls. Only valid characters should be received by the program (G, 1, 2, 3, 4). If the user enters '-1', the program should be able to handle such input. It should not crash. Do not use GOTO statements.

Problem 11.c

Write a **C programme** that does the following:

1. asks the user to enter two numbers R and C where the values of R and C are less than 10;
2. asks the user to enter the elements of the matrix (of order R rows * C columns) and prints it out on the screen;
3. asks the user to enter the elements of the matrix (of order C rows * R columns) and prints it out on the screen;
4. multiplies the two matrices and displays the result on the screen.

Also check whether a user has provided integers for R and C and if they are a positive number and less than 10. If the input values for R and C are not positive integer and are not lower than 10, print out the error message.

Example 1:

```
Enter two positive numbers R and C: 2 3
Enter elements of the first matrix (2 x 3):
Enter element e11: 1
Enter element e12: 2
Enter element e13: 3
Enter element e21: 4
Enter element e22: 5
Enter element e23: 6

Entered first matrix:
1  2  3
4  5  6

Enter elements of the second matrix (3 x 2):
Enter element e11: 7
Enter element e12: 8
Enter element e21: 9
Enter element e22: 10
Enter element e31: 11
Enter element e32: 12

Entered second matrix:
 7   8
 9  10
11  12

Result of multiplication (2 x 2 matrix):
 58   64
139  154
```

Problem 12.c

A program that will allow the user to enter a number from 0 to 1000. The number should be converted to correct its Roman Numeral Format. Hint: I, V, X, L, C, D, M

Problem 13.c

Write a C program to input elements in an array. After all the numbers are inserted in the array, put even and odd elements in two separate arrays.

Example:

Input:

Input the size of an array: 10

Elements to be inserted in the array: 0 1 2 3 4 5 6 7 8 9

Output:

Array of the even elements: 0 2 4 6 8
Array of the odd elements: 1 3 5 7 9

Problem 14.c

Write a program in C to count a total number of duplicate elements in an array. Minimum should be 5 and maximum is 25 elements. Inputs must be integers (both negative and positive).

Problem 15.c

Create a C program to find the length of the longest substring of a given string without repeating characters

```
Original String: xyzxyzyy
Length of the longest substring without repeating characters: 3
```

Problem 16.c

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13. The first six prime numbers are 2, 3, 5, 7, 11, and 13. Write a C programming to compute for the 1001st prime number?

END OF HOMEWORK. Goodluck!