Online Match Finder

Online dating websites are big business venture nowadays. The lure of these websites is that one can find a potential match from around the world. Complex algorithms are often employed by these websites, based on user profiles, to determine compatibility. You have been hired by one such startup company as a rookie programmer. The situation is that the company owners highly believe in numerology. You have been assigned to the numerical compatibility module. You are to write a program to perform the following task(s):

(1) Read user data from a file provided to your program at runtime named users.txt. Each line of the file contains user data with the following format:

FirstName LastName Gender [M/F] *DateofBirth* [DDMMYYYY]

Examples: John Doe M 14111985
Amy Ann F 07091993

First and last name will contain only upper and lower-case letters from the English alphabet; each name will be at most 10 letters. The four fields will be separated by a space.

(2) Calculate the Characteristic Adequateness of all individuals/users in the input file following the algorithm given below:

Step 1: Calculate the Numerical Weight of the name (W_{name}).

Each letter of the first and last name have a value representing their 1-based index in the alphabet (e.g., A=1, B=2, c=3, D=4). The case of the letter does not change its value. You may use the tolower() or toupper() functions, which change the case of a character, by including the cctype header (i.e., #include <cctype>).

To compute W_{name},multiply the value of each letter as described above by its index in the user's first name. Do the same for the user's last name. Finally, the summation of the two values from the first and last name is W_{name}.

Step 2: Calculate the Numerical Weight of the D.O.B. (W_{dob}). The W_{name} is the summation over each of the digits in the user's DateofBirth.

Step 3: Calculate Numerical Adequateness(N_{adq}). $N_{adq} = W_{name} * W_{dob}$

Example :

J o h n D o e

Weight of Name [John Doe] → **(10*1 + 15*2 + 8*3 + 14*4) + (4*1 + 15*2 + 5*3) = 169**

Weight of D.O.B → **(1 + 4 + 1 + 1 + 1 + 9 + 8 + 5) = 30.**

N_{adq} → 169 * 30 = 5070

Step 4: Compute all proper divisors of the Numerical Adequateness from the above step. Next, compute the sum of these proper divisors. A description of proper divisors is given at the end of this description. Finally, compare this sum with the Numerical Adequateness (N_{adq}) to determine the user's category as shown below:

If **SumOfProperDivisors(**N_{adq}) $< N_{adq}$, the individual is characteristically deficient,
$= N_{adq}$, the individual is characteristically perfect,
$> N_{adq}$, the individual is characteristically abundant.

Example:

SumOfProperDivisors(5070) = 8106, hence John Doe is characteristically abundant.

(3) Two users of opposite sexes are considered compatible except in the following pairs:

(a) deficient – deficient
(b) deficient – perfect (or perfect - deficient).

Generate all pairs that are compatible and store each pair in the output file *match_results.txt*, each on a separate line in the following format:

MaleUser FemaleUser

Notes:

(i) Consider pairs of only the opposite sexes.
(ii) Generate each pair only once.
(iii) Use a separate routine to calculate the SumOfProperDivisors( ).
(iv) The file *users.txt* will contain data for at most 50 users. I will test your code against various input files. I suggest you do the same.

[Concepts Tested: File I/O, Arithmetic, Arrays, Strings, Booleans, Loops, Comparison etc.]

Proper Divisor (mathematical concept used in this assignment):

A proper divisor, of a number *n*, is any positive integer that evenly divides *n* and is less than *n*. For example:

The divisors of the number 30 are 1, 2, 3, 5, 6, 10, 15, 30 (as they divide 30 without remainders). But the **proper divisors** of 30 are 1, 2, 3, 5, 6, 10, 15. The number 30 is not considered a proper divisor of itself.

Pseudocode:

The term pseudocode is often used to describe an algorithm (sequence of steps to accomplish the task at hand) using typical programming language constructs alongside text to ease readability and understandability of an algorithm.

It is generally good practice to create a solution written in pseudocode before writing code in a specific language (e.g., C++, Java).

Pseudocode for the function ***SumOfProperDivisors(n)*** is listed below:

function ***SumOfProperDivisors***(Number ***n***) :

initialize variable ***sum*** **:= 1**;

for ***i*** from **2** to ***n*****/2** do the following step

if ***i*** evenly divides ***n*** then ***sum*** **:=** ***sum*** **+** ***i*** ;

function returns **sum**;

Grading Criteria:

- The program compiles. If the program does not compile no further grading can be accomplished. Programs that do not compile will receive a zero.
- (15 Points) The program executes without exception and produces output. The grading of the output cannot be accomplished unless the program executes.
- (60 Points) Proper File Operations, calculations performed & correct pairs are detected & outputted.
- (5 Points) The program is documented (e.g., commented) properly.
- (5 Points) Proper indentation.
- (5 Points) Descriptive and consistent naming standards followed.
- (10 Points) Variable declarations
    - Variables' scope is reasonably constrained (e.g., your program should not have global variables)
    - Type of each variable expresses how it is used (e.g., use constant when value of variable should not change)