Intelligent Clinical Impression

Submission Date: **June 17 12noon**

A **clinical impression** is referred to as the initial opinion of a doctor after examining a patient. The doctor can give several impressions on a patient's condition. Once the patient has undergone further examinations, the doctor can give a diagnosis which is considered the final opinion of the doctor about the patient's disease.

Clinical diagnoses and impressions may be generated by an intelligent computer system. In one study in Indiana University, a computer program has shown to be 42% better than human in terms of diagnosing and treating health conditions. Artificial intelligence techniques were used in the system.

The goal of this project is to develop a similar program that can generate a list of **clinical impressions** along with other patient information in a summary in the form of history of present illness (HPI). HPI can help doctors in diagnosing a patient.

Below are the requirements that you need to accomplish to be able to create an *intelligent* clinical impression system. The objectives of this project are to use the appropriate data structure in storing doctor and patient information and to be able to access and manipulate them properly and efficiently.

For this project, you are required to use arrays, structures, strings and text files. There are two kinds of users – the Doctor and the Patient.

PROGRAM REQUIREMENTS:

The program should start with the display of Menu #1.

```
MENU #1 User Type

D for Doctor
P for Patient
E for Exit

Enter your choice: D
```

- Choices are **D**, **P** and **E**
- If the choice is **D**, follow the steps in Part I and show Menu #2 Doctor
- If the choice is **P**, follow all the steps in Part II. This will work only if there is an existing list of symptoms and impressions.
- If the choice is **E**, exit the program.

Part I. Building of Dataset from Doctor (User: Doctor)

Your task is to store into a file the important information on certain diseases and their symptoms. Such information will be provided by the doctor and includes a list of clinical impressions, symptoms and their corresponding questions.

Below is the Doctor Menu.

```
MENU #2 Doctor

C to Create a new list of symptoms and impressions
U to Use the existing list of symptoms and impressions
D to Display Symptoms
M to Modify Symptoms
E to Exit

Enter your choice: C
```

- Choices are **C**, **U**, **D**, **M** and **E**

- If the choice is **C**, follow the tasks indicated in **I1** to **I5** (Input Symptoms and Impressions) and store the list of symptoms as well as the list of impressions in a file.
- If the choice is **U**, proceed to Main Menu (Menu #1 User Type). The assumption here is that there is an existing file of symptoms and impressions that was previously created by a doctor. The symptoms and impressions must be read from the file and copied to a data structure.
- If the choice is **D**, follow **D1-D2 (Display Symptoms)**
- If the choice is **M**, follow **M1-M2 (Modify Symptoms)**
- If the choice is **E**, return to Menu #1 User Type. Before the program exits, all the symptoms and impressions must be stored in a file. The design of the file is left to the programmer.

For the Patient's Reading of Symptoms as in Part II, there is an **additional requirement** - the program should be able to generate a text file for the HPI of a patient after asking for the symptoms. The file must be named using the surname of the patient.

---------- **Option I to INPUT SYMPTOMS AND IMPRESSIONS** ----------

I1: Ask for the number of symptoms (let's say this is **N**). Maximum value for N is 20..

```
How many symptoms do you want to consider?  10
```

I2: Enter the name of each symptom and its corresponding question. Do this for **N** times and store the two entries in a data structure. Choose the appropriate data structure for such information.

```
Symptom # 1:
What is the symptom?  High Fever
How do you want to ask the symptom?
Is your body temperature 39 degrees or above? (Y/N)


Symptom # 2:
What is the symptom?  Body Pain
How do you want to ask the symptom?
Is your entire body in pain? (Y/N)

Symptom # 3:
What is the symptom?  Headache
How do you want to ask the symptom?
Do you have a headache? (Y/N)

:


:
Symptom # 10:
What is the symptom?  Body Rashes
How do you want to ask the symptom?
Does your entire body have rashes? (Y/N)
```

I3: Ask for the number of impressions (let's say this is **I**). Maximum value for I is 20.

```
How many impressions do you want to enter? 20
```

I4: For **I** times, ask for each impression and list all the symptom names entered in I2.

```
Impression # 1:
What is the illness?  Flu
Below is a list of symptoms.

1. High Fever
2. Body Pain
```

```
3. Headache
   :
10. Body Rashes

How many of the symptoms above are present in a Flu case? 3
Enter the corresponding number of each symptom:
1
2
3

Impression # 2
What is the illness?  Dengue
Below is a list of symptoms.

1. High Fever
2. Body Pain
3. Headache

   :
10. Body Rashes

How many of the symptoms above are present in a Dengue case? 4
Enter the corresponding number of each symptom:
1
2
3
10

:
Impression # 10
What is the illness?  High Blood Pressure
:
```

I 5: Return to the **Main Menu.**

---------- **Option D to DISPLAY SYMPTOMS** ----------

D1: Ask for the Impression and then display the corresponding symptoms.

```
What is the Impression? Flu
Symptoms of a Flu are:
      High Fever
      Body Pain
Headache
```

D2: Return to the **Main Menu**

---------- **Option M to MODIFY SYMPTOMS** ----------

This option allows the doctor to modify the symptoms of an impression.

M1: Ask for the Impression and display the current symptoms.

```
What is the Impression? Flu
Symptoms of Flu are:
      High Fever
      Body Pain
Headache
```

M2: Display the list of symptoms (like in I4) then ask for the number of symptoms and which symptoms are present in this particular disease/impression.

```
You can modify the symptoms of Flu.
Below is a list of symptoms.
```

```
      1. High Fever
      2. Body Pain
      3. Headache
         :
  9. Nausea
      10. Body Rashes

 How many of the symptoms above are present in a Flu case? 4
 Enter the corresponding number of each symptom:
      1
      2
      3
      9

 Here are the NEW SYMPTOMS of Flu:
      High Fever
      Body Pain
Headache
      Nausea
```

Part II. Reading of symptoms from Patient to generate a History of Present Illness (User: Patient)

Once all the impression and symptoms are entered in Part I and stored in the computer's memory (initially then later into a file), a patient will be asked for some personal details and different symptoms that the he/she is experiencing, to generate a History of Present Illness.

Step 1: Enter patient information

```
What is your name? Juan dela Cruz
What is your patient number? 1234
Enter your age:  42
Gender (M/F): M
```

Step 2: Ask for the symptoms using the questions entered in Step #2. Note that if the answer is either 'N' or 'n', then the symptom is not observed or present.

Step 3: Display the History of Present Illness (HPI) with patient details, symptoms and impressions.

```
Is your body temperature 39 degrees or above? (Y/N)   Y
Is your entire body in pain? (Y/N)   Y
Do you have a headache? (Y/N)   Y
:
Does your entire body have rashes? (Y/N)   Y


            History of Present Illness

Juan dela Cruz, patient # 1234 is a 42-year old male. He has high fever, body
pain, headache and body rashes. Impressions are Flu and Dengue.
```

The impressions are Flu and Dengue since the first 3 symptoms that are associated with them are reported by the patient. Note that if there are more than 1 symptom/impression, use separators like "**,**" and "**and**".

Step 4: Return to the **Main Menu.**

Notes on Data Structures.

Some user inputs must initially be stored in a data structure then later into a file.

String size – maximum size is 50.

Text Files format (see sample text files in File Folder for MP)

Symptoms.txt format

<number of symptoms>
<symptom number – starts at 1 >
<Symptom name>
<Question>
<symptom number >
<Symptom name>
<Question>
:
<symptom number >
<Symptom name>
<Question>

Impressions.txt format

<number of impressions>
<Impression number – starts at 1>
<Impression name>
<List of Symptoms in numbers >
<Impression number >
<Impression name>
<List of Symptoms in numbers >
:
<Impression number >
<Impression name>
<List of Symptoms in numbers >

History of present illness - <patient Number>.txt
Ex. 1234.txt if a patient has a number 1234, based on the example above

```
History of Present Illness

Juan dela Cruz, patient # 1234 is a 42-year old male. He has high fever, body
pain, headache and body rashes. Impressions are Flu and Dengue.
```

Bonus
A **maximum** of **10 points** may be given for features **over & above** the requirements, like (1) producing top 2 of the languages when the program uses a mix of languages (not necessarily with equal count); (2) allowing user input of more than one word for the translations and processes different features like identifying the language and the translation considering multiple words; or other features not conflicting with the given requirements or changing the requirements) subject to **evaluation** of the teacher. **Required features** must be **completed first** before bonus features are credited. Note that use of conio.h, or other advanced C commands/statements may **not** necessarily merit bonuses.

Submission & Demo

- **Final MP Deadline: June 17, 2022 (F), 1200noon via Canvas**. After the indicated time, the submission page will remain open for submission until June 20, 1200noon only but for every day late, there will be a 20% deduction in the MP grade. Note that any amount of time (even a few seconds after 12noon of June 17) will already be considered the next day. Thus, submissions from June 17 12:01PM to June 18 12:00PM (noon) will incur 20% deductions; submissions from June 18, 12:01PM to June 19, 12:00PM will incur 40% deductions; submissions from June 19, 12:01PM until June 20, 12PM will incur 60% deductions in the MP grade. At June 20, 12:01PM, the submission page will be locked and thus considered no submission (equivalent to 0 in the MP grade).

  Requirements: Complete Program
  - Make sure that your implementation has considerable and proper use of arrays, strings, structures, files, and user-defined functions, as appropriate, even if it is not strictly indicated.
  - It is expected that each feature is supported by at least one function that you implemented. Some of the functions may be reused (meaning you can just call functions you already implemented [in support] for other features. There can be more than one function to perform tasks in a required feature.
  - Debugging and testing was performed exhaustively. The program submitted has
    a. NO syntax errors
    b. NO warnings - make sure to activate -Wall (show all warnings compiler option) and that C99 standard is used in the codes

c. NO logical errors -- based on the test cases that the program was subjected to

IMPLEMENTATION, DOCUMENTATION & TESTING REQUIREMENTS:

1. Use **gcc -Wall** to compile your C program. Make sure you **test** your program completely (compiling & running).
2. Do not use brute force. Use **appropriate conditional** statements **properly**. Use, **wherever appropriate**, **appropriate loops** & **functions properly**.

3. You **may** use topics outside the scope of CCPROG2 but this will be **self-study**. Goto *label*, exit(), break (except in switch), continue, global variables, calling main() are **not allowed**.

4. Include **internal documentation** (comments) in your program.

5. The following is a checklist of the deliverables:

Checklist:
- Upload via AnimoSpace submission:
  - source code*
  - test script**
  - sample text file exported from your program
- email the softcopies of all requirements as attachments to **YOUR** own email address on or before the deadline

Legend:

***** Source code exhibit readability with supporting inline documentation (not just comments before the start of every function) and follows a coding style that is similar to those seen in the course notes and in the class discussions. The first page of the source code should have the following declaration (in comment) [replace the pronouns as necessary if you are working with a partner]:

```
/*********************************************************************************************
This is to certify that this project is my own work, based on my personal efforts in studying and
applying the concepts learned.  I have constructed the functions and their respective algorithms
and corresponding code by myself.  The program was run, tested, and debugged by my own efforts.  I
further certify that I have not copied in part or whole or otherwise plagiarized the work of other
students and/or persons.

               <your full name>, DLSU ID# <number>
*********************************************************************************************/
```

Example coding convention and comments before the function would look like this:

```
/* funcA returns the number of capital letters that are changed to small letters
   @param strWord - string containing only 1 word
   @param pCount - the address where the number of modifications from capital to small are
                   placed
   @return 1 if there is at least 1 modification and returns 0 if no modifications
   Pre-condition: strWord only contains letters in the alphabet
*/
int     //function return type is in a separate line from the
funcA(char strWord[20] ,    //preferred to have 1 param per line
      int * pCount)            //use of prefix for variable identifiers
{  //open brace is at the beginning of the new line, aligned with the matching close brace
   int    ctr;        /* declaration of all variables before the start of any statements -
                         not inserted in the middle or in loop- to promote readability */

   *pCount = 0;
   for (ctr = 0; ctr < strlen(strWord); ctr++)  /*use of post increment, instead of pre-
                                                  increment */
   {  //open brace is at the new line, not at the end
      if (strWord[ctr] >= 'A' && strWord[ctr] <= 'Z')
      {    strWord[ctr] = strWord[ctr] + 32;
           (*pCount)++;
      }
      printf("%c", strWord[ctr]);
   }

   if (*pCount > 0)
      return 1;
```

```
    return 0;
}
```

***Test Script should be in a table format. There should be at least 3 categories (as indicated in the description) of test cases **per function**. There is no need to test functions which are only for screen design (i.e., no computations/processing; just printf).

Sample is shown below.

Function	**#**	**Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
sortIncreasing	1	Integers in array are in increasing order already	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	P
	2	Integers in array are in decreasing order	aData contains: 53 37 33 32 15 10 8 7 3 1	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	P
	3	Integers in array are combination of positive and negative numbers and in no particular sequence	aData contains: 57 30 -4 6 -5 -33 -96 0 82 -1	aData contains: -96 -33 -5 -4 -1 0 6 30 57 82	aData contains: -96 -33 -5 -4 -1 0 6 30 57 82	P

Test descriptions are supposed to be unique and should indicate classes/groups of test cases on what is being tested. Given the sample code in page 6, the following are four distinct classes of tests:

i.) testing with strWord containing all capital letters (or rephrased as "testing for at least 1 modification")
ii.) testing with strWord containing all small letters (or rephrased as "testing for no modification")
iii.) testing with strWord containing a mix of capital and small letters
iv.) testing when strWord is empty (contains empty string only)

The following test descriptions are **incorrectly formed**:

- Too specific: testing with strWord containing "HeLlo"
- Too general: testing if function can generate correct count OR testing if function correctly updates the strWord
- Not necessary -- since already defined in pre-condition: testing with strWord containing special symbols and numeric characters

6. Upload the softcopies via Submit Assignment in Canvas. You can submit multiple times prior to the deadline. However, only the last submission will be checked. Send also to your **mylasalle account** a **copy** of all deliverables.

7. Use **<surnameFirstInit>.c** & **<surnameFirstInit >.pdf** as your filenames for the source code and test script, respectively. You are allowed to create your own modules (.h) if you wish, in which case just make sure that filenames are descriptive.

8. During the MP **demo**, the student is expected to appear on time, to answer questions, in relation with the output and to the implementation (source code), and/or to revise the program based on a given demo problem. Failure to meet these requirements could result to a grade of 0 for the project.

9. It should be noted that during the MP demo, it is expected that the program can be compiled successfully and will run. If the program does not run, the grade for the project is automatically 0. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) and other submissions (e.g., non-violation of restrictions evident in code, test script, and internal documentation) will still be checked.

10. The MP should be an HONEST intellectual product of the student/s. For this project, you are allowed to do this individually or to be in a group of 2 members only. Should you decide to work in a group, the following mechanics apply:

Individual Solution: Even if it is a group project, each student is still required to create his/her INITIAL solution to the MP individually without discussing it with any other students. This will help ensure that each student went through the process of reading, understanding, solving the problem and testing the solution.

Group Solution: Once both students are done with their solution: they discuss and compare their respective solutions (ONLY within the group) -- note that learning with a peer is the objective here -- to see a possibly different or better way of solving a problem. They then come up with their group's final solution -- which may be the solution of one of the students, or an improvement over both solutions. Only the group's final solution, with internal documentation (part of comment) indicating whose code was used or was it an improved version of both solutions) will be submitted as part of the final deliverables. It is the group solution that will be checked/assessed/graded. Thus, only 1 final set of deliverables should be uploaded by one of the members in the Canvas submission page. [Prior to submission, make sure to indicate the members in the group by JOINing the same group number.]

Individual Demo Problem: As each is expected to have solved the MP requirements individually prior to coming up with the final submission, both members should know all parts of the final code to allow each to INDIVIDUALLY complete the demo problem within a limited amount of time (to be announced nearer the demo schedule). This demo problem is given only on the day/time of the demo, and may be different per member of the group. Both students should be present/online during the demo, not just to present their individual demo problem solution, but also to answer questions pertaining to their group submission.

Grading: the MP grade will be the same for both students -- UNLESS there is a compelling and glaring reason as to why one student should get a different grade from the other -- for example, one student cannot answer questions properly OR do not know where or how to modify the code to solve the demo problem (in which case deductions may be applied or a 0 grade may be given -- to be determined on a case-to-case basis).

11. Any form of **cheating** (**asking other people not in the same group for help**, **submitting as your [own group's] work part of other's work**, **sharing your [individual or group's] algorithm and/or code to other students not in the same group, etc.**) can be punishable by a grade of **0.0** for the **course** & a **discipline case**.

Any requirement not fully implemented or instruction not followed will merit deductions.