Assignment #1

Date Due: November 1, 2022

Total: 100 marks

Inter Process Communication

You need to create a client-server program for each of the protocols listed below for processing marks for at least 3 courses and at least 5 students in each course. The client handles the interaction with the user, and the server performs all computations.

- Input (handled by the client program only): a list of courses and marks obtained by at least 5 students for assignments, exams, and other tests. The marking scheme, as well as the marks, and students should be read for each course.
- Output:
  - The Server program (after receiving relevant data from the client) will produce a file with the name `CoursesHighestMarks.txt`, containing the names of the courses, the highest mark in that course, the name of the student, and the marks.
  - Client program: the name of the file produced by the server, the names of the courses, the name of the student with the corresponding highest marks, and the last confirmation from the server (please see below).

Example:

The client accepts as input the following information (the exact format and how you type it at the standard output is your choice)

```
CS3520 Operating Systems
5
Student1 10 20 30 50 60
Student2 10 20 0 50 60
Student3 10 50 30 50 60
Student4 10 30 30 50 60
Student5 10 20 40 50 60
3
20 30 50
```

(first three are assignments, next midterm and final, Assignments 20% Midterm 30% final 50%),

```
CS3610  Analysis and Design of Algorithms
5
Student1 100 20 30 50 60
```

```
Student2 10 20 0 50 60
Student3 10 50 30 50 60
Student4 10 30 30 50 60
Student5 90 80 70 50 60
3
30 30 40
```

(first three are assignments, next midterm and final, Assignments 30% Midterm 30% final 40%),

```
CS2620  Comparative Programming languages
6
Student1 100 20 30 50 60 20
Student2 10 20 0 50 60 0
Student3 10 50 30 50 60 100
Student4 10 30 30 50 60 90
Student5 90 80 70 50 60 78
Student6 50 40 30 60 70 80
4
45 20 35
```

(first four are assignments, next midterm and final, Assignments 45% Midterm 20% final 35%), then you read this information using the client program, and send it to the server program.

The server program will then compute the final mark for each course, and each student that is

$$\frac{(10+20+30)}{3}\cdot 0.20 + 50\cdot 0.3 + 60\cdot 0.5 = 49$$

$$\frac{(10+20+0)}{3}\cdot 0.20 + 50\cdot 0.3 + 60\cdot 0.5 = 47$$

$$\frac{(10+50+30)}{3}\cdot 0.20 + 50\cdot 0.3 + 60\cdot 0.5 = 51$$

$$\frac{(10+30+30)}{3}\cdot 0.20 + 50\cdot 0.3 + 60\cdot 0.5 = 49.6$$

$$\frac{(10+20+40)}{3}\cdot 0.20 + 50\cdot 0.3 + 60\cdot 0.5 = 49.6$$

for CS3520,

$$\frac{(100+20+30)}{3}\cdot 0.3 + 50\cdot 0.3 + 60\cdot 0.4 = 54$$

$$\frac{(10+20+0)}{3}\cdot 0.3 + 50\cdot 0.3 + 60\cdot 0.4 = 42$$

$$\frac{(10+50+30)}{3}\cdot 0.3 + 50\cdot 0.3 + 60\cdot 0.4 = 40$$

$$\frac{(10+30+30)}{3}\cdot 0.3 + 50\cdot 0.3 + 60\cdot 0.4 = 45.9$$

$$\frac{(90+80+70)}{3}\cdot 0.3 + 50\cdot 0.3 + 60\cdot 0.4 = 63$$

for CS3610, and

$$\frac{(50+40+30+60)}{4}\cdot 0.45+70\cdot 0.2+80\cdot 0.35=62.25$$
$$\frac{(50+0+30+60)}{4}\cdot 0.45+70\cdot 0.2+80\cdot 0.35=57.75$$
$$\frac{(50+40+80+60)}{4}\cdot 0.45+70\cdot 0.2+80\cdot 0.35=67.65$$
$$\frac{(40+40+30+60)}{4}\cdot 0.45+70\cdot 0.2+80\cdot 0.35=66.75$$
$$\frac{(50+40+90+60)}{4}\cdot 0.45+70\cdot 0.2+80\cdot 0.35=69$$
$$\frac{(0+80+30+60)}{4}\cdot 0.45+70\cdot 0.2+80\cdot 0.35=60.90$$

for CS2620.

The file written by the server `CoursesHighestMarks.txt` will contain the lines[1]:

```
Operating Systems mark 51 Student3 10 50 30 50 60
Analysis and Design of Algorithms mark 63 Student5 90 80 70 50 60
Comparative Programming languages mark 69 Student5 90 80 70 50 60 78
```

The information sent back to the client would be the acronim of the course the highest mark the name of the student having the highest mark (all on one line separated by spaces):

```
CS3520 51 Student3 CS3610  63 Student5   CS2620 69 Student5
```

The client program will present this information to the user (standard output), and the user will then type a new course number together with its name. The client will receive this information and append a new line to the file just created. So, in case the user types

```
MCS3320 Theory of Computing,
```

the server will append that input to the file `CoursesHighestMarks.txt`, and its final content will be:

```
Operating Systems mark 51 Student3 10 50 30 50 60
Analysis and Design of Algorithms mark 63 Student5 90 80 70 50 60
Comparative Programming languages mark 69 Student5 90 80 70 50 60 78
MCS3320 Theory of Computing
```

The server will then confirm the successful writing and send this information to the client. The client will acknowledge the user by printing a corresponding message to the standard output.

The program should be written as a client-server setup, where the server process does all the computations, and the client process interacts with the user(s).

Requirements:

[1] As mentioned above, it will contain on each line, the name of the course, followed by `mark`, then the highest mark, and the name of the student having the highest mark followed by the list of marks

- The program that accepts the input and displays the output, i.e., the interface program, should not do any other computation than communicating with a server program. Your client program should accept input from the programmer and send it to the server program. It receives the result from the server and prints it.
- The server program will perform all the computations and send the results back to the interface program. Your server program will accept connections and compute final marks for each course.
- The type of communication between the programs(client and server) will form a complete suite.
- A default marking scheme for computing your mark may be stored on the server. If you accept a flexible marking scheme(a variable number of tests, for example), you add 5 marks to the communication protocol.

  You have the freedom to choose the syntax of your files (standard input/standard output), but it must be in a human-readable format. The exact syntax for each file should be well documented and included in a Readme file. Compiling should be done using an appropriate make file.

  Each suite is marked as follows:

  1. client program 5 marks
  2. server program 5 marks
  3. communication protocol (explained and properly implemented) 5 marks (additional 5 if marking scheme is flexible)

- The communication protocol is one of the following:
    1. named pipes (fixed names, one for client to server, the other one for server to client)
    2. unnamed pipes
    3. named sockets (one for each connection)
    4. internet sockets on localhost
    5. named shared memory
    6. private shared memory.

For each program (20 marks): the Readme file, Makefile, and proper documentation will be worth 5 marks (if not present, the mark cannot be more than 15 for that suite). Testing programs worth another 3 marks (for each suite). These are additional marks on top of the 20. Showing an execution of your programs should be done to get these points (text files only, no png/gif/jpeg/pdf pictures).

All programs should work on our Virtual machines (provided). If they do not work on the Virtual machines but are working in another environment, that will `NOT` be counted as a good program. Marking is only done if the program works on the Virtual machine.