Course	COMP-8567
Instructor	Dr. B. Boufama
Project	Due on April 19, 2020, (**to be demonstrated to the tutors**)

Note: This project is to be done by pairs (two students) or by one student

Write two programs, a client and a server, to implement a simple online game. The server process and the client process will run on two different machines and the communication between the two processes is achieved using Sockets.

The server/client tasks can be summarized as follows :

- The server must start running before any client, and goes into an infinite loop to wait for clients.
- When the server gets a client, it waits for another client (two players are needed)
- When the server gets the other client (now two clients), it forks and, let the child process take care of these two clients (players) in a separate function, called **servicePlayers**, while the parent process goes back to wait for the next two clients (players). Note that the file descriptors of the two connected sockets (one for each client) are passed to the function **servicePlayers**. If you prefer, you can also use threads instead of forking.
- The server's child process will be a referee for the two clients, call them TOTO and TITI, who would be the two players in a scenario similar to example 4, from chapter VII (slide 8). In particular, the referee first gets in an infinite loop, then
  1. it sends the message "You can now play" to player TOTO,
  2. it reads the scrore obtained by TOTO and adds it to TOTO's total,
  3. it sends the message "You can now play" to player TITI,
  4. it reads the scrore obtained by TITI and adds it to TITI's total,
  5. If one or both accumulated scores (totals) reaches 100, the referee
     - sends "Game over: you won the game" to the winner and sends "Game over: you lost the game" to the player who lost,
     - closes both sockets (each player has its own socket) and exits,
  6. otherwise, goes back to step 1.
- Once a client (player) is connected to the server, it gets in an infinite loop then,
  1. it reads the server's message (waits for the message),
  2. if the read message is "You can now play", client plays its dice, prints obtained score on its screen and sends it to the server, before it goes to step 1.
  3. if the read message is "Game over: you won the game", player prints "I won the game" then closes socket and exit.
  4. if the read message is "Game over: you lost the game", player prints "I lost the game" then closes socket and exit.