**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
CSC 457  
Assignment 1 – Processes**50 points

**Assigned**: 9/26/2022  
**Due:** 10/5/2022

**Evaluation:**

1. Program runs without crashing (\_\_\_\_\_ / 10)
2. PCB includes all relevant fields with appropriate data types (\_\_\_\_\_ / 15)
3. PCBs are loaded correctly and displayed (\_\_\_\_\_ / 15)
4. Prompt for priority level and display corresponding PCBs (\_\_\_\_\_ / 10)

***Note: poor indentation, commenting, etc. will result in a lower grade.***

**ASSIGNMENT**:

* Complete this assignment in C. Submit your C file to Canvas. It must be able to run in Visual Studio Code, under the configuration we installed in class.
* You are provided with two data files. Instead of explicitly opening a file, redirect standard input on the command line so that you can simply use scanf.
  + PowerShell is the terminal you’re using in Visual Studio Code. In PowerShell, you can direct input to the program as follows:
    - get-content FILENAME.DAT | .\MYPROGRAM
    - Where FILENAME.DAT is the data file and MYPROGRAM is the name of your executable. You must run this from the folder where the executable and the data file are located.
* You may work in pairs or alone. If working in a pair, you must be present when the work is submitted, and it is your responsibility to make sure your name is on the submitted work.
* You are writing a (console) program in order to create an array of (simulated, fake, simplified) process control blocks. The program should read from a file (processes.txt), one line for each process control block (format given below). You can assume that there will be no more than 600 when creating the array in this program. After all input is read in, show all processes. Then prompt the user for a priority level and output all processes with that priority level, and also a count of how many of those processes there are.
* Each line of the file has the following (separated by a space) (I am providing two files of processes, one with 40, one with 500. The one with 40 should be plenty for testing this program)
  + int processId;
  + int userId;
  + int arrivalTime; // time units in simulation, not actual time
  + int priority; // base priority
  + int expectedTimeRemaining;
  + int expectedMemoryNeed;
  + int expectedPctCPU; // to get an idea of whether CPU bound or IO bound
  + bool realTime; // 0 or 1 - whether a real-time process or not (real-time processes may need immediate attention)
* Create a struct to represent the PCB. Include all of the above information PLUS the following, which is not part of the input file:
  + State processState; // Not input - initially NEW
  + int currentPriority; // can be increased or lowered based on what has happened with the process - not input - initially same as base priority
  + int timeWaitingSoFar; // Not input - initially zero
  + int timeProcessingLastRun; // Not input - initially zero
  + int timeProcessingSoFar; // Not input - initially zero
  + struct processblock\* nextPtr; // not used in this program
* Use functions to divide up the task. Use the following functions:
  + A function to read a line from the file and fill up a PCB
  + A function to find and display all PCBs with a given priority level
  + A function to display one PCB with the following format:

ID: 3 Usr: 3 Arr: 0 pri: 1 Remain: 1 Mem: 17 CPU: 10