Pair Programming 4 Activities

* **Always use the pair programming tests to ensure your program works properly. Evaluation is based primarily upon correct execution. Activities without test screen shots and/or code will be earned. Both must be included.**
* **Take a screen shot with a white background of each execution in the tests.**
* **Download the source code file for inclusion in the turn in document.**
* **Turn in pair programming activities using the pair programming turn in document.**
* **It is each individual’s responsibility to turn in the assignment and pair programming is graded individually so make sure you share the work you and your partner did together as you go.**
* **Do not share work with your partner that you did not do together.**
* **Pair Programming is group work, but you can only work with your assigned partner. If you do not work with your partner, you can only earn 50% of pair programming points.**
* **Make sure you have your partner’s name, username and contact information such as Pellissippi WebMail.**

4a. (5 points) From your home directory, create a directory called pp4a with the command  
mkdir pp4a  
Change to that directory with the command  
cd pp4a  
Create the following files then execute and test your program.

* queueItem.h (similar to stackItem.h) with a typedef for a QueueItem
* queue.h (similar to stack.h) with constants, a typedef for the Queue structure and prototypes for functions. See the Lecture Notes for further instructions and help with the pair programming.
* queue.c: has the function definitions of the function prototypes in queue.h
* mainQueue.c: copy mainQueue.c from ~caarnold/cisp1020/week4 and use it to test your queue.
* Makefile

See the Lecture Notes for further instructions and help with the pair programming.

Use the pair programming test cases to test your program. Make corrections if the program output does not match the tests. Capture screen shots of each correct execution.

4b. (5 points) From your home directory, create a directory called pp4b with the command  
mkdir pp4b  
Change to that directory with the command  
cd pp4b  
Copy ~caarnold/cisp1020/week4/binaryReadStart.c to binaryRead.c and complete it so it reads an unknown number of Employee structures (at most 100, though) into an array of Employee structures using the sample input file binary.out (also in ~caarnold/cisp1020/week4) whose name is retrieved from the command line. Make sure there is a command line argument and that the input file exists before reading. The structures read have to be \*exactly\* like the structures written by binaryWrite.c discussed in the Lecture Notes So, you \*must\* read entire structures at a time (not individual data members) and use the exact same structure declaration as the Employee structure in binaryWrite.c. The first name declared first with space for 30 characters. The last name declared second, age (an int) next then salary (a double) declared last. The binary data in this file was written in this format. Even if a name didn’t actually take up 30 characters, 30 characters were written. After reading in the data from the binary file, write it out to stdout so you can see if your reading was successful.

See the Lecture Notes for further instructions and help with the pair programming.

Use the pair programming test cases to test your program. Make corrections if the program output does not match the tests. Capture screen shots of each correct execution.