Pair Programming 5 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.**

5a. (5 points) From your home directory, create a directory called pp5a with the command  
mkdir pp5a  
Change to that directory with the command  
cd pp5a  
Copy ~caarnold/cisp1020/week5/gradesStart.c and update it to ask the user how many grades he/she has, dynamically allocate an array of that many. Then, get that many grades from the user and put them into your dynamically allocated array. Calculate the average and print

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.

5b. (5 points) From your home directory, create a directory called pp5b with the command  
mkdir pp5b  
Change to that directory with the command  
cd pp5b  
Copy ~caarnold/cisp1020/week5/mainDblLinkedList.c and read through the file then use it to compile and test the files below. If you decide to test functions as you write them, you will need to comment out some code in main that uses those functions you haven’t written yet, or you can just write stubs for some functions. Create C and header files for the doubly linked list and node. The files to create are:

* node.h: contains the type definition for a node and any function declarations/prototypes for a node. Don’t forget to use the #ifndef NODE\_H #define NODE\_H #endif pre-processor directives.
* node.c: contains all node function definitions that were declared in node.h. This file must #include “node.h” since it has code that references the node\_t structure which is in node.h.
* dbl\_linked\_list.h: this file contains the structure definition for the list. It also contains function declarations/prototypes for all list functions
* dbl\_linked\_list.c,

It is very important that the correct definitions and declarations are put in the correct files. All functions that operate only on nodes such as initNode \*must\* go in the node files. All functions that operate on the entire list such as insertNode which inserts in a list or findNode which finds a node in a list \*must\* go in the list files. If a function has code that operates on a list at all, it \*must\* go in the list files, not the node files.

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.