Pair Programming 8 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.**

8a. (5 points) This program requires the creation and use of a linked list using classes in C++. Using Codelite, create a workspace called pp8 and create a project called pp8a. Copy the contents of the main8a.cpp file from the Pair Programming 8 Assignment and paste it into the main.cpp file in the Codelite project. Add the files and add the code to them referred to in the Lecture Notes. This code should include the following.

* book.h, book.cpp
* node.h, node.cpp. node.h is shown below.

|  |
| --- |
| #ifndef NODE\_H  #define NODE\_H  #include "book.h"  class Node {  private:  Book book;  Node\* nextPtr;  Node\* prevPtr;  public:  Node();  Node( const Book& );  Node( const Node& );  virtual ~Node() {}  void setBook( const Book& );  Book getBook() const { return book; }  // LinkedList is a friend  friend class LinkedList;  };  #endif |

* linkedlist.h, linkedlist.cpp. linkedlist.h is shown below.

|  |
| --- |
| #ifndef LINKEDLIST\_H  #define LINKEDLIST\_H  #include "book.h"  #include "node.h"  class LinkedList {  public:  LinkedList() : headPtr( NULL), tailPtr( NULL ), count( 0 ) {}  LinkedList( const LinkedList& );  void insertNode( Node\* ); // at head  void traverseQueue() const;  void traverseStack() const;  Node\* findNode( const Book& ) const;  void deleteNode( Node\* );  Node\* dequeueNode(); // from tail  Node\* popNode(); // from head  virtual ~LinkedList() { deleteList(); }  void deleteList();  bool isEmpty() const { return headPtr == NULL; }  LinkedList& operator=( const LinkedList& );  private:  Node\* headPtr;  Node\* tailPtr;  int count;  };  #endif |

Add a new file, b1.txt, to the project and copy the text in b1.txt in the Pair Programming 8 Assignment into this new file. When you’re ready to execute the program, copy b1.txt to this project’s Debug folder (which will not exist until successful compilation) then execute the project. Cleaning or rebuilding (which cleans and builds) the project will remove the Debug folder, so if you clean and/or rebuild, you will have to recopy b1.txt to the Debug folder.

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.

8b. (5 points) This program creates a binary file called books.bin, reads from it, and prints the books read from the binary file to standard output to verify the binary writing and reading work. Add another project to the pp8 workspace called pp8b. Copy the code in the Pair Programming 8 Assignment’s main8b.cpp into this project’s main.cpp. Add two new files to the project (DON’T ADD EXISTING FILES) book.h and book.cpp and copy the code from these pp8a files to the pp8b files. Add code to the Book class to complete pp8b which includes:

* ostream& write( ostream& ) const;
* istream& read( istream& );

Put b1.txt in the Pair Programming 8 Assignment in this project’s Debug folder (which will not exist until successful compilation) then execute the project. Cleaning or rebuilding (which cleans and builds) the project will remove the Debug folder, so if you clean and/or rebuild, you will have to recopy b1.txt to the Debug folder.

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.