C++ MATH TUTOR FINAL GRADE SHEET
DELIVERABLE #3 Due 11/13/14
Include a printed copy of this sheet with submission.

PROGRAMMER LAST NAME_____

Total possible points for each item appear in **bold**.	**SCORE**
1. Hardcopy Source Code for OOP Program - The code must be well-structured/designed and formatted, with comments sufficient to show the intent of all code for someone attempting to maintain it. Must be easily readable, with all classes easy to find. Must be complete.	**20**
2. Hardcopy Source Code contains - 1) At least 3 classes; 2) The User Interface is encapsulated so all interaction goes through it; 3) The program will accommodate up to 100 student records.	**15**
3. Hardcopy Run Output 1 - A run of the program for Student #1 showing the selection of Random Math Operations for the easy 1 digit problems. Should show the 10 problems and answers, with some correct and some incorrect answers.	**15**
4. Hardcopy Run Output 2 - A run of the program for Student #2 showing the selection of Random Math Operations for the most difficult problem option. Should show the 10 problems and answers, with some correct and some incorrect answers.	**15**
5. Hardcopy Run Output 3 - A run of the program showing the student result data for at least Student #1 and #2, from the previous runs above.	**10**
6. Test Code Source and Output Sample for at least 2 Unit Tests - Provides source for a main program(s) that tests at least two of the functions from two different classes. Includes sample output for the unit tests, demonstrating correct function.	**15**
7. Thumb Drive Containing ONLY Windows EXE File- Program runs without problems. User interface is clear, easy to use for elementary level student.	**25**
8. EXE Runs according to original requirements	**20**
9. Presentation- This submission has no hand-written parts (except labels); submitted in a folder; has well-organized, readable pages, within which it is **easy to find each of the elements** listed on this sheet; includes a printout of this sheet for the grader. Design and Test Plan are NOT to be resubmitted.	**15**
TOTAL OUT OF 150 Comments:	

```cpp
#include <stdio.h>
#include <String>
#include <iostream>
#include <time.h>
#include <cstdlib>
#include <cmath>
#include "progress.h"
#include "question.h"
#include "ui.h"
#include "test_question.h"
#include "test_progress.h"
#include "test_ui.h"
#include "test.h"

int main(int argc) {

    if (argc == 2) {
        // Check the number of arguements.  If there is more than one, test mode

        test testing; // Create a new testing object

        // Execute the tests on our Testing object
        testing.executeTests();

        // Pause so we can see the output
        system("PAUSE");
    }
    else {
        ui mainUI;               // Create a UI object
        progress students[100]; // Student progress tracker
        int stu = 0;             // The number of students already stored
        question questions;      // Create a question object
        int menuChoice = 1;      // The choice the user makes on a menu
        int userAnswer = 0;      // The user's answer to a question
        bool secondTry = false; // Whether this is their second try
        int i = 0;               // Loop counter variable

        while (menuChoice != 0) {
            // While it's not the menu choice to exit the program

            // Show our lovely header
            mainUI.genHeader();

            // Show our main menu
            mainUI.genMainMenu();

            // Ask the user what they want to do
            menuChoice = mainUI.getAnswer();

            if (menuChoice == 1) {
                // If it's the menu choice for answering questions

                // Intitialize a student object
```

```
54                  students[stu].setStudentName(mainUI.askStudentName());
55                  questions.setLevel(mainUI.askLevel());
56                  questions.setOp(mainUI.askOperation());
57                  students[stu].activate();
58
59                  // Generate a new question
60                  questions.generateQuestion();
61
62                  // Ask the user ten questions
63                  while (i < 10) {
64                      mainUI.askQuestion(questions.getNumOne(),
65                          questions.getNumTwo(), questions.getOp(), secondTry);
66
67                      if (questions.checkAnswer(mainUI.getAnswer())) {
68                          // If the answer is true, increment counter and
69                          // generate a new question
70                          students[stu].answerTrue();
71                          questions.generateQuestion();
72                          secondTry = false;
73                          mainUI.answerEchoResult(true);
74                          i++;
75                      }
76                      else {
77                          // If the answer is false, increment counter and
78                          // check to see if the student is on their second try...
79                          students[stu].answerFalse();
80                          if (secondTry) {
81                              // ... If so, new question
82                              questions.generateQuestion();
83                              secondTry = false;
84                              i++;
85                          }
86                          else {
87                              // ... If not, flag it so it is a second try
88                              secondTry = true;
89                          }
90
91                          // Output to the user
92                          mainUI.answerEchoResult(false);
93                      }
94                  }
95
96                  // Reset the counter and increment the number of students
97                  i = 0;
98                  stu++;
99              }
100             else if (menuChoice == 2) {
101                 // If it's the menu choice for seeing answers, output them
102                 for (int i = 0; students[i].getActive(); i++) {
103                     mainUI.echoStudentScores(students[i]);
104                 }
105             }
106             else if (-1 > menuChoice || menuChoice > 3) {
```

```
107                     // Invalid menu choice, let the user know
108                     mainUI.answerChoiceNotVaild();
109                     mainUI.genHeader();
110                     mainUI.genMainMenu();
111                 }
112                 else {
113                     // Do nothing, the loop will terminate
114                 }
115             }
116         }
117 
118         // Tell the OS we're done and happy :)
119         return 0;
120     }
```

```
1   using namespace std;
2
3   class progress {
4   private:
5       bool active;          // Whether this class is active
6       int numTrue;          // Number of true answers
7       int numFalse;         // Number of false answers
8       string studentName;   // Student's name
9   public:
10      progress();
11      void answerTrue();
12      void answerFalse();
13      void resetStats();
14      int getNumTrue();
15      int getNumFalse();
16      string getStudentName();
17      void setStudentName(string);
18      bool getActive();
19      void activate();
20  };
21
22  progress::progress() {
23      // Constructor: Sets the variables
24      // Requrements: N/A
25      numTrue = 0;
26      numFalse = 0;
27      studentName = "";
28      active = false;
29  }
30  void progress::answerTrue() {
31      // Increments the internal true counter
32      // Requrements: 210
33      numTrue++;
34  }
35
36  void progress::answerFalse() {
37      // Increments the internal false counter
38      // Requrements: 190, 210
39      numFalse++;
40  }
41
42  void progress::resetStats(){
43      // Reset the stats for the new user
44      // Requrements: 250, 330
45      numTrue = 0;
46      numFalse = 0;
47      studentName = "";
48
49  }
50
51  int progress::getNumTrue(){
52      // Returns the number of answers that the user has
53      // answered right
```

```
54      // Requirements: 220, 340
55      return numTrue;
56  }
57  int progress::getNumFalse() {
58      // Returns the number of answers that the user has
59      // answered wrong
60      // Requirements: 220, 340
61      return numFalse;
62  }
63
64  string progress::getStudentName() {
65      // Returns the student's name
66      // Requirements: N/A
67      return studentName;
68  }
69
70  void progress::setStudentName(string tempName) {
71      // Sets the student's name
72      // Requirements: N/A
73      studentName = tempName;
74  }
75
76  bool progress::getActive() {
77      // Returns whether the class is active
78      // Requriements: N/A
79      return active;
80  }
81
82  void progress::activate() {
83      // Activates this class, should be called after values are set
84      // Requirements: N/A
85      active = true;
86  }
```

```
using namespace std;

class question {
    private:
        int randomNumber(bool, bool, int);
        int generateAnswer();
        int numOne;
        int numTwo;
        int answer;
        int level;
        int op;
        bool useTempOp;
        int tempOp;
        int prevNumOne;
        int prevNumTwo;
    public:
        question();
        void generateQuestion();
        bool checkAnswer(int);
        void setLevel(int);
        void setOp(int);
        int getOp();
        int getNumOne();
        int getNumTwo();

};

int question::randomNumber(bool randOp = false, bool division = false, int part = 0) {
    // Wrapper for the random function, allows the
    // creation of requirement-fitting random numbers
    // Requirements: 110, 260, 270, 290, 300

    int bottomValue;
    int topValue;

    if (randOp) {
        bottomValue = 1;
        topValue = 4;
    }
    else if (division) {
        if (part == 1) {
            if (level == 2 || level == 3) {
                topValue = 99;
                bottomValue = 1;
            }
            else {
                topValue = 9;
                bottomValue = 1;
            }
        }
        else if (part == 2) {
            if (level == 3) {
                topValue = 99;
```

```
                bottomValue = 1;
            }
            else {
                topValue = 9;
                bottomValue = 1;
            }
        }
        else {
            topValue = 9;
            bottomValue = 1;
        }
    }
    else {
        switch (level) {
        case 1:
            bottomValue = 1;
            topValue = 9;
            break;
        case 2:
            bottomValue = 1;
            topValue = 99;
            break;
        case 3:
            bottomValue = 1;
            topValue = 999;
            break;
        default:
            bottomValue = 1;
            topValue = 1;
            break;
        }
    }

    return bottomValue + (rand() % topValue);

}

int question::generateAnswer() {
    int localOp;                // Temporary storage of the operator

    if (useTempOp) {
        localOp = tempOp;
    }
    else {
        localOp = op;
    }

    switch (localOp) {
    case 1:
        return numOne + numTwo;
        break;
    case 2:
        return numOne - numTwo;
```

```
107             break;
108         case 3:
109             return numOne * numTwo;
110             break;
111         case 4:
112             return numOne / numTwo;
113             break;
114         default:
115             return 0;
116             break;
117         }
118     }
119
120     question::question() {
121         srand(time(NULL));
122     }
123
124     void question::generateQuestion() {
125         // Generates a new question for the student to work on
126         // Requrements: 120, 200, 260, 280
127         int tempNumOne;                         // Temporarily generated number 1
128         int tempNumTwo;                         // Temporarily generated number 2
129         int localOp = randomNumber(true);       // Random operator, used if op == 5
130
131         // Division checking and generating our initial set of numbers
132         if (op == 4) {
133             tempNumOne = randomNumber(false, true, 1);
134             tempNumTwo = randomNumber(false, true, 2);
135         }
136         else if (op == 5 && localOp == 4) {
137             tempNumOne = randomNumber(false, true, 1);
138             tempNumTwo = randomNumber(false, true, 2);
139         }
140         else {
141             tempNumOne = randomNumber();
142             tempNumTwo = randomNumber();
143         }
144
145         // If the first number matches a previous number, generate a new one
146         while (tempNumOne == prevNumOne || tempNumOne == prevNumTwo) {
147             if (op == 4) {
148                 tempNumOne = randomNumber(false, true, 1);
149             }
150             else if (op == 5 && localOp == 4) {
151                 tempNumOne = randomNumber(false, true, 1);
152             }
153             else {
154                 tempNumOne = randomNumber();
155             }
156         }
157
158         // If the second number matches a previous number, generate a new one
159         while (tempNumTwo == prevNumTwo || tempNumTwo == prevNumOne) {
```

```
160             if (op == 4) {
161                 tempNumTwo = randomNumber(false, true, 2);
162             }
163             else if (op == 5 && localOp == 4) {
164                 tempNumTwo = randomNumber(false, true, 2);
165             }
166             else {
167                 tempNumTwo = randomNumber();
168             }
169         }
170 
171         // If the two numbers match, generate a new one for the second number
172         while (tempNumOne == tempNumTwo) {
173             tempNumTwo = randomNumber();
174             while (tempNumOne == prevNumOne || tempNumOne == prevNumTwo) {
175                 if (op == 4) {
176                     tempNumTwo = randomNumber(false, true, 2);
177                 }
178                 else if (op == 5 && localOp == 4) {
179                     tempNumOne = randomNumber(false, true, 2);
180                 }
181                 else {
182                     tempNumTwo = randomNumber();
183                 }
184             }
185             while (tempNumTwo == prevNumTwo || tempNumTwo == prevNumOne) {
186                 if (op == 4) {
187                     tempNumTwo = randomNumber(false, true, 2);
188                 }
189                 else if (op == 5 && localOp == 4) {
190                     tempNumTwo = randomNumber(false, true, 2);
191                 }
192                 else {
193                     tempNumTwo = randomNumber();
194                 }
195             }
196         }
197 
198         // Checking to see which number is on top
199         if (tempNumOne < tempNumTwo) {
200             numOne = tempNumTwo;
201             numTwo = tempNumOne;
202         }
203         else {
204             numOne = tempNumOne;
205             numTwo = tempNumTwo;
206         }
207 
208         // Save the variables, set a flag if we're using a temporary operator
209         if (op != 5) {
210             useTempOp = false;
211             if (op == 4) {
212                 numOne = numOne * numTwo;
```

```
213             }
214             answer = generateAnswer();
215             tempOp = op;
216         }
217         else {
218             useTempOp = true;
219             tempOp = localOp;
220             if (localOp == 4) {
221                 numOne = numOne * numTwo;
222             }
223             answer = generateAnswer();
224             op = 5;
225         }
226 
227         // store the previous variables for later use
228         prevNumOne = numOne;
229         prevNumTwo = numTwo;
230     }
231 
232     bool question::checkAnswer(int testValue) {
233         // Checks the answer against the currently stored
234         // question
235         // Requrements: 140, 190
236         return testValue == answer;
237     }
238 
239     void question::setLevel(int tempLevel) {
240         // Changes the level of the questions generated
241         // Requrements: 260, 270, 300
242         level = tempLevel;
243     }
244 
245     void question::setOp(int localOp) {
246         // Changes the operator used
247         // Requrements: N/A
248         op = localOp;
249     }
250 
251     int question::getOp() {
252         // Returns the operator used for generating the question
253         // Includes checking to see if we're using a temporarily stored op
254         // Requriements: N/a
255         if (useTempOp) {
256             return tempOp;
257         }
258         else {
259             return op;
260         }
261     }
262 
263     int question::getNumOne() {
264         // Returns the first number of the question
265         // Requirements: 110
```

```
266         return numOne;
267     }
268 
269     int question::getNumTwo() {
270         // Returns the second number of the question
271         // Requirements: 110
272         return numTwo;
273     }
```

```
using namespace std;

class ui {
private:
    void errorMessage(string);
public:
    const string SCHOOLNAME = "University of Colorado Colorado Springs";
    const string TAB = "     ";
    void genHeader();
    void genMainMenu();
    void clearConsole();
    string askStudentName();
    int askOperation();
    int askLevel();
    void askQuestion(int, int, int, bool);
    int getAnswer();
    void echoStudentScores(progress); // progress[] & );
    void answerEchoResult(bool);
    void answerChoiceNotVaild();
};

void ui::errorMessage(string message) {
    // Helper function that allows us to produce consistant error messages
    // Requirements: N/A
    cout << "An error has occured in the program." << endl;
    cout << "The error condition reported: " << message << endl;
    cout << "You can try that action again, but you may get the same result :(";
    cout << endl;
    cout << endl;
}

void ui::genHeader() {
    // Generates a welcome and instructions for the
    // user
    // Requirements: N/A

    cout << "Welcome to the Computer Aided Instruction System" << endl;
    cout << "This software licensed to: " << SCHOOLNAME << endl;
    cout << endl;
    cout << endl;
}
void ui::genMainMenu() {
    // Generates the main menu
    // Requirements: N/A
    cout << "Please enter your menu choice: " << endl;
    cout << TAB << "1 - Answer questions" << endl;
    cout << TAB << "2 - View Student scores" << endl;
    cout << TAB << "0 - exit" << endl;
}

void ui::clearConsole() {
    // Clears the console using a while loop and endl
    // Requirements: N/A
```

```
54          for (int i = 0; i < 80; i++) {
55              cout << endl;
56          }
57      }
58      string ui::askStudentName() {
59          // Asks for the student's name and returns it in the
60          // form "<first name> <last name>"
61          // Requirements: 330
62          string first; // First Name
63          string last;  // Last Name
64
65          // Ask for the names
66          cout << "Please enter your first name: ";
67          cin >> first;
68          cout << "Please enter your last name: ";
69          cin >> last;
70
71          // Combine and return
72          return first + " " + last;
73      }
74
75      int ui::askOperation() {
76          // Asks which operation the student would like to do
77          // Reuqirements: N/A
78          int tempAnswer; // The answer given by the user
79
80          // Output the options to the screen
81          cout << "Please enter the operation you would like to do:" << endl;
82          cout << TAB << "1 - Addition" << endl;;
83          cout << TAB << "2 - Subtraction" << endl;
84          cout << TAB << "3 - Multiplication" << endl;
85          cout << TAB << "4 - Division" << endl;
86          cout << TAB << "5 - A mixture of all four" << endl;
87
88          // Get the user's answer
89          tempAnswer =  getAnswer();
90
91          // Some error checking
92          while (tempAnswer < 1 || tempAnswer > 5) {
93              errorMessage("That is an invalid menu choice.");
94              tempAnswer = getAnswer();
95          }
96
97          // Return the user's choice
98          return tempAnswer;
99      }
100
101     int ui::askLevel() {
102         // Asks the user which level the want to do
103
104         int tempAnswer; // The answer give by the user
105
106         // Output the option to the screen
```

```
107         cout << "Please enter which level you would like to try (1-3)" << endl;
108
109         // Get the user's answer
110         tempAnswer = getAnswer();
111
112         // Some error checking
113         while (tempAnswer < 1 || tempAnswer > 3) {
114             errorMessage("That is an invalid menu choice.");
115             tempAnswer = getAnswer();
116         }
117
118         // Return their choice
119         return tempAnswer;
120     }
121
122     void ui::askQuestion(int numOne, int numTwo, int op, bool secondTry = false) {
123         // Asks the student a question
124         // Inputs: first number, second number, second attempt
125         // Requirements: 120
126         string textOperator; // A textual representation of the operator
127
128         // Populate the textOperator field
129         switch (op) {
130         case 1:
131             textOperator = " + ";
132             break;
133         case 2:
134             textOperator = " - ";
135             break;
136         case 3:
137             textOperator = " * ";
138             break;
139         case 4:
140             textOperator = " / ";
141             break;
142         default:
143             textOperator = " ? ";
144             break;
145         }
146
147         // If this is the user's second try, output a notation to that effect
148         if (secondTry) {
149             cout << "(Second attempt) ";
150         }
151
152         // Output the actual problem
153         cout << numOne << textOperator << numTwo << " = " << endl;
154     }
155
156
157     int ui::getAnswer() {
158         // Gets the answer from a student using cin
159         // Returns the answer given
```

```
160         // Requirements: 130
161         int temp; // The answer given
162
163         cout << "Enter your answer: ";
164
165         cin >> temp;
166
167         return temp;
168     }
169     void ui::echoStudentScores(progress studentProgress) {
170         // Outputs the student scores from each item of the
171         // array to the screen
172         // Requirements: 220, 230, 240, 330, 340
173         cout << "Displaying scores for: " << studentProgress.getStudentName() << endl;
174         cout << "Number correct: " << studentProgress.getNumTrue() << endl;
175         cout << "Number wrong: " << studentProgress.getNumFalse() << endl;
176
177         // Let the user know if they're ready to go on to the next level
178         if ((studentProgress.getNumTrue() /
179             (studentProgress.getNumTrue() + studentProgress.getNumFalse()))
180             > 75 / 100) {
181             cout << studentProgress.getStudentName() << " is ready to go on to the"
182                 << " next level" << endl;
183         }
184
185         // new line, for luck
186         cout << endl;
187     }
188     void ui::answerEchoResult(bool resultStatus) {
189         // Outputs the text indicating the result to the user
190         // Input: Whether the answer is true or false
191         // Requirements: 150, 160, 170, 180
192         int number;
193
194         string right[4] = { "Very good!", "Excellent!", "Nice work!",
195          "Keep up the good work!" }; // Collection of phrases for right
196         string wrong[4] = { "No. Please try again!", "Wrong. Try once more.",
197          "Don't give up!", "No. Keep trying." }; // Collection of phrases for wrong
198
199         // Random number for which one to use
200         number = rand() % 4;
201
202         // Output to the screen
203         if (resultStatus) {
204             cout << right[number] << endl;
205         }
206         else {
207             cout << wrong[number] << endl;
208         }
209     }
210
211
212     void ui::answerChoiceNotVaild() {
```

```
213         // Outputs an error message to the user.
214         // Requirements: N/A
215         errorMessage("That menu choice is not valid ");
216     }
217
```

```
using namespace std;

class test {
private:
    test_question tQuest;                // Temporary testing question object
    test_progress tProg;                 // Temporary testing progres object
    test_ui tUI;                         // Temporary testing UI iobject
    void generateClassHeading(string);
public:
    void executeTests();

};

void test::generateClassHeading(string filename) {
    // Visual element, generates boxed headings for each class
    // Requirements: N/A

    // Get the length of the string, add 4 for the boxes
    int c = filename.length() + 4;

    // Some padding
    cout << endl;
    cout << endl;
    cout << endl;

    // Output the top of the box
    for (int i = 0; i <= c; i++) {
        cout << "-";
    }

    // Output a newline and then the file name with sides
    cout << endl;
    cout << "| " << filename << " |" << endl;

    // Output the bottom of the box
    for (int j = 0; j <= c; j++) {
        cout << "-";
    }

    // One more line break, for luck
    cout << endl;
}

void test::executeTests() {
    bool finalTestStatus_questions = true;  // Whether the question tests passed
    bool finalTestStatus_progress = true;   // Whether the progress tests passed
    bool finalTestStatus_ui = true;         // Whether the ui tests passed

    // Output the mode of the program, to avoid confusion
    cout << "Tests executing..." << endl;

    // Running tests on question.h
    generateClassHeading("question.h");
```

```
54        finalTestStatus_questions = tQuest.executeTests();
55
56        // Running tests on progress.h
57        generateClassHeading("progress.h");
58        finalTestStatus_progress = tProg.executeTests();
59
60        // Running tests on ui.h
61        generateClassHeading("ui.h");
62        finalTestStatus_ui = tUI.executeTests();
63
64        // Output the final test status of question.h
65        cout << "Final Test Status (Question class): ";
66        if (finalTestStatus_questions) {
67            cout << "Succeeded!" << endl;
68        }
69        else {
70            cout << "Failed :(" << endl;
71        }
72
73        // Output the final test status of progress.h
74        cout << "Final Test Status (Progress class): ";
75        if (finalTestStatus_progress) {
76            cout << "Succeeded!" << endl;
77        }
78        else {
79            cout << "Failed :(" << endl;
80        }
81
82        // Output the final test status of ui.h
83        cout << "Final Test Status (UI class): ";
84        if (finalTestStatus_ui) {
85            cout << "Succeeded!" << endl;
86        }
87        else {
88            cout << "Failed :(" << endl;
89        }
90    }
```

```
1   using namespace std;
2
3   class test_progress {
4   private:
5       progress mainProgress;  // Testing progress variable
6       bool setValues();
7       bool getValues();
8       void breakline();
9   public:
10      bool executeTests();
11  };
12
13  void test_progress::breakline() {
14      // Visual function, outputs an 80 character breakline
15      // Requriements: N/A
16
17      // Output a newline for ease of use
18      cout << endl;
19
20      // Create 80 dashes
21      for (int i = 0; i <= 80; i++) {
22          cout << "-";
23      }
24
25      // Another newline, for luck
26      cout << endl;
27  }
28
29  bool test_progress::setValues() {
30      bool returnVal = true;  // Whether the test succeeded
31
32      try {
33          // Attempt to set the values of the class
34          mainProgress.activate();
35          mainProgress.setStudentName("Matthew Bowker");
36          mainProgress.answerFalse();
37          mainProgress.answerFalse();
38          mainProgress.answerTrue();
39          mainProgress.answerTrue();
40          mainProgress.answerFalse();
41      }
42      catch (...) {
43          // If it fails, the test fails
44          returnVal = false;
45      }
46
47      // Return whether the test succeeded
48      return returnVal;
49  }
50
51  bool test_progress::getValues() {
52      bool returnValue = true; // Whether the test succeed
53
```

```
54      try {
55          // Output the stored values from the module
56          cout << "Is the module active? " << mainProgress.getActive() << endl;
57          cout << "Student Name: " << mainProgress.getStudentName() << endl;
58          cout << "Number true: " << mainProgress.getNumTrue() << endl;
59          cout << "Number false: " << mainProgress.getNumFalse() << endl;
60      }
61      catch (...) {
62          // If it fails, the test fails
63          returnValue = false;
64      }
65
66      // Return whether he test succeeded
67      return returnValue;
68  }
69
70  bool test_progress::executeTests() {
71      bool returnValue = true;
72
73      cout << "Getting values for an inactive module..." << endl;
74      if (getValues() == false) {
75          returnValue = false;
76      }
77
78      breakline();
79
80      cout << "Setting values and activating module..." << endl;
81      if (setValues() == false) {
82          returnValue = false;
83      }
84
85      breakline();
86
87      cout << "Getting values for an active module..." << endl;
88      if (getValues() == false) {
89          returnValue = false;
90      }
91
92      return returnValue;
93  }
```

```
using namespace std;

class test_question {
private:
    question mainQuestion;                  // Test instace of the Question obj
    bool test_function_generateQuestion(); // Test function 1
    bool test_function_getAnswer();        // Test function 2
    const string TAB = "     ";            // Constant to help align output
public:
    bool executeTests();                    // External API to run the tests
};

bool test_question::test_function_generateQuestion() {
    // Function generates 10 questions at each level of the program,
    // checks them against the given parameters, and then repeats.

    // Functions tested: generateQuestion(), getNumOne(), getNumTwo()
    bool returnValue = true; // Whether the test succeeded
    int level = 1;           // Starting level
    int prevNumOne = 0;      // The first number from the previous problem
    int numOne = 0;          // The first number from the current problem
    int prevNumTwo = 0;      // The second number from the previous problem
    int numTwo = 0;          // The second number from the current problem


    while (level <= 3) {
        // Testing each level individually using this loop

        // Output the level so we know which one we're doing
        cout << "Level " << level << ": " << endl;

        // Set the level in our test class
        mainQuestion.setLevel(level);

        for (int i = 0; i < 10; i++) {
            // Loop to generate ten questions to test

            // Generate the question then output the count
            mainQuestion.generateQuestion();
            cout << "Question " << i + 1 << " Generated." << endl;

            // Retrieve the stored values
            numOne = mainQuestion.getNumOne();
            numTwo = mainQuestion.getNumTwo();

            // Compare the stored values against my three comparison tests.
            // If one fails, it sets returnValue to false
            if (prevNumOne == numOne) {
                cout << TAB;
                cout << "Two numbers were generated in a row for the first "
                    << "number. The numbers " << numOne << " and "
                    << prevNumOne << " matched. " << endl;
                returnValue = false;
```

```
54              }
55              else if (prevNumTwo == numTwo) {
56                  cout << TAB;
57                  cout << "Two numbers were generated in a row for the second"
58                      << " number. The numbers " << numTwo << " and "
59                      << prevNumTwo << " matched. " << endl;
60                  returnValue = false;
61              }
62
63              if (numOne == numTwo) {
64                  cout << TAB;
65                  cout << "The two numbers matched: "
66                      << numOne << " and " << numTwo << "." << endl;
67                  cout << endl;
68                  returnValue = false;
69              }
70
71              // Store the numbers for the next test.
72              prevNumOne = numOne;
73              prevNumTwo = numTwo;
74          }
75
76          // Increment the level counter
77          level++;
78      }
79
80      // Return the success condition
81      return returnValue;
82  }
83
84  bool test_question::test_function_getAnswer() {
85      // Function generates 10 questions at each level of the program,
86      // Answers them, then checks the answers with the ones stored in the class
87
88      // Functions tested: generateQuestion(), getNumOne(), getNumTwo(),
89      // setLevel(), setOp(), getOp(), checkAnswer()
90
91      bool returnValue = true; // Bool to store if the test passed
92      int numOne;              // Storage variable for the fist number
93      int numTwo;              // Storage variable for the second number
94      int level = 1;           // Level of the problem.  Starting at 1
95      int tempAnswer;          // The answer this function generates
96      int tempRemainder;       // A remainder for divison problems
97      int op;                  // Integer representation of the operator
98      string opVisual;         // Visual representation of the operator
99
100     // Setting the intial level so this first generateQuestion function
101     // doesn't die a horrible death
102     mainQuestion.setLevel(level);
103
104     // Generate our first question
105     mainQuestion.generateQuestion();
106
```

```
107         for (op = 1; op <= 5; op++) {
108             // Loop through each operator, setting it then testing it.
109             mainQuestion.setOp(op);
110 
111             for (level = 1; level <= 3; level++) {
112                 // Loop through all the levels for each operator
113 
114                 // Output which level we're on
115                 cout << "Level " << level << ":";
116 
117                 // Set the level for the test class
118                 mainQuestion.setLevel(level);
119 
120                 // Generate our question
121                 mainQuestion.generateQuestion();
122 
123                 // Retrieve the numbers from the class
124                 numOne = mainQuestion.getNumOne();
125                 numTwo = mainQuestion.getNumTwo();
126 
127                 // Generate a local answer to comapre against
128                 switch (op) {
129                 case 1:
130                     tempAnswer = numOne + numTwo;
131                     opVisual = " + ";
132                     break;
133                 case 2:
134                     tempAnswer = numOne - numTwo;
135                     opVisual = " - ";
136                     break;
137                 case 3:
138                     tempAnswer = numOne * numTwo;
139                     opVisual = " * ";
140                     break;
141                 case 4:
142                     tempAnswer = numOne / numTwo;
143                     tempRemainder = numOne % numTwo;
144                     opVisual = " / ";
145                     break;
146                 case 5:
147                     switch (mainQuestion.getOp()) {
148                     case 1:
149                         tempAnswer = numOne + numTwo;
150                         opVisual = " + ";
151                         break;
152                     case 2:
153                         tempAnswer = numOne - numTwo;
154                         opVisual = " - ";
155                         break;
156                     case 3:
157                         tempAnswer = numOne * numTwo;
158                         opVisual = " * ";
159                         break;
```

```
160                 case 4:
161                     tempAnswer = numOne / numTwo;
162                     tempRemainder = numOne % numTwo;
163                     opVisual = " / ";
164                     break;
165                 }
166             }
167 
168             // If the operation is division, check for a remainder
169             // then output to the screen
170             if (mainQuestion.getOp() == 4) {
171                 cout << numOne << opVisual << numTwo << " = " << tempAnswer
172                     << " (Remainder " << tempRemainder << ")";
173             }
174             else {
175                 cout << numOne << opVisual << numTwo << " = " << tempAnswer;
176             }
177 
178             // If the stored answer matches the class' answer, succeed
179             // Else set the success condition to false
180             if (mainQuestion.checkAnswer(tempAnswer)) {
181                 cout << " --- Matches!" << endl;
182             }
183             else {
184                 cout << " --- Does not Match :(" << endl;
185                 returnValue = false;
186             }
187 
188         }
189     }
190 
191     // Return the success condition
192     return returnValue;
193 }
194 
195 bool test_question::executeTests() {
196     bool returnValue = true; // Whether the tests succeeded
197 
198     // Output a header so we know what's going on.
199     cout << "test_function_generateQuestion" << endl;
200     cout << "------------------------------" << endl;
201 
202     // Execute the generateQuestion test
203     // If it fails set the success flag to false
204     if (test_function_generateQuestion() == false) {
205         returnValue = false;
206     }
207 
208     // Some blank lines, yay!
209     cout << endl;
210     cout << endl;
211 
212     // Another heading
```

```
213         cout << "test_function_getAnswer" << endl;
214         cout << "----------------------" << endl;
215 
216         // Execute the getAnswer test
217         // If it fails, set the success flag to false
218         if (test_function_getAnswer() == false) {
219             returnValue = false;
220         }
221 
222         // Return whether the function succeeded
223         return returnValue;
224     }
```

```
using namespace std;

class test_ui {
private:
    ui mainUI;           // The testing UI object
    bool testOutputs();
    void breakline();
public:
    bool executeTests();
};

void test_ui::breakline() {
    // Visual function, outputs an 80 character breakline
    // Requriements: N/A

    // Output a newline for ease of use
    cout << endl;

    // Create 80 dashes
    for (int i = 0; i <= 80; i++) {
        cout << "-";
    }

    // Another newline, for luck
    cout << endl;
}
bool test_ui::testOutputs() {
    bool tempReturn = true;       // Whether the run succeeded
    string askStudentName_output; // The output of askStudentName()
    int getAnswer_output;         // The output of getAnswer()
    progress testProgress;        // Testing progress object

    // Set the values of our testing progress object
    testProgress.setStudentName("Charlie Test");
    testProgress.answerTrue();
    testProgress.answerTrue();
    testProgress.answerTrue();
    testProgress.answerFalse();
    testProgress.answerFalse();

    try {
        // Attempt to generate the proper output

        // Test function genHeader()
        cout << "Header generation" << endl;
        cout << "-----------------" << endl;
        mainUI.genHeader();
        breakline();

        // Test function genMainMenu()
        cout << "Main menu generation" << endl;
        cout << "--------------------" << endl;
        mainUI.genMainMenu();
```

```
54          breakline();
55
56          // Test function askStudentName()
57          cout << "Ask student name" << endl;
58          cout << "----------------" << endl;
59          askStudentName_output = mainUI.askStudentName();
60          cout << "The student name entered was: " << askStudentName_output << endl;
61          breakline();
62
63          // Test function askLevel()
64          cout << "Asking the level of the questions" << endl;
65          cout << "---------------------------------" << endl;
66          mainUI.askLevel();
67          breakline();
68
69          // Test function askOperation()
70          cout << "Asking the operation" << endl;
71          cout << "--------------------" << endl;
72          mainUI.askOperation();
73          breakline();
74
75          // Test function askQuestion()
76          cout << "Asking a question the first time" << endl;
77          cout << "---------------------------------" << endl;
78          mainUI.askQuestion(10, 5, 2, false);
79          breakline();
80
81          // Test function AskQuestion() (again)
82          cout << "Asking a question the second time" << endl;
83          cout << "---------------------------------" << endl;
84          mainUI.askQuestion(10, 5, 2, true);
85          breakline();
86
87          // Test function getAnswer()
88          cout << "Getting an answer" << endl;
89          cout << "-----------------" << endl;
90          getAnswer_output = mainUI.getAnswer();
91          cout << "The answer entered was: " << getAnswer_output << endl;
92          breakline();
93
94          // Test function echoStudentScores()
95          cout << "Outputting student scores" << endl;
96          cout << "-------------------------" << endl;
97          mainUI.echoStudentScores(testProgress);
98          breakline();
99
100         // Test function answerEchoResult() (true)
101         cout << "Student answered question right" << endl;
102         cout << "-------------------------------" << endl;
103         mainUI.answerEchoResult(true);
104         breakline();
105
106         // Test function answerEchoResult() (false)
```

```
107             cout << "Student answered question wrong" << endl;
108             cout << "------------------------------" << endl;
109             mainUI.answerEchoResult(false);
110             breakline();
111
112             // Test function answerChoiceNotValid()
113             cout << "Student's choice is invalid" << endl;
114             cout << "----------------------------" << endl;
115             mainUI.answerChoiceNotVaild();
116             breakline();
117         }
118         catch (...) {
119             // If it breaks, tests failed
120             tempReturn = false;
121         }
122
123         // Return whether we succeeded
124         return tempReturn;
125     }
126
127     bool test_ui::executeTests (){
128         // API to allow the running of tests
129         bool tempReturn = true;      // Whether the tests succeeded
130         tempReturn = testOutputs(); // Execute the test
131         return tempReturn;          // Return whether it worked
132     }
```

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

F:\>FinalProject.exe
Welcome to the Computer Aided Instruction System
This software licensed to: University of Colorado Colorado Springs


Please enter your menu choice:
     1 - Answer questions
     2 - View Student scores
     0 - exit
Enter your answer: 1
Please enter your first name: Easy
Please enter your last name: Student
Please enter which level you would like to try (1-3)
Enter your answer: 1
Please enter the operation you would like to do:
     1 - Addition
     2 - Subtraction
     3 - Multiplication
     4 - Division
     5 - A mixture of all four
Enter your answer: 5
8 * 6 =
Enter your answer: 48
Nice work!
5 + 2 =
Enter your answer: 7
Excellent!
9 * 8 =
Enter your answer: 17
No. Keep trying.
(Second attempt) 9 * 8 =
Enter your answer: 72
Excellent!
6 - 4 =
Enter your answer: 3
No. Keep trying.
(Second attempt) 6 - 4 =
Enter your answer: 2
Nice work!
63 / 7 =
Enter your answer: 9
Nice work!
9 * 3 =
Enter your answer: 27
Very good!
16 / 2 =
Enter your answer: 8
Nice work!
7 - 1 =
Enter your answer: 6
```

```
Excellent!
8 + 6 =
Enter your answer: 14
Keep up the good work!
14 / 2 =
Enter your answer: 6
Wrong. Try once more.
(Second attempt) 14 / 2 =
Enter your answer: 7
Very good!
Welcome to the Computer Aided Instruction System
This software licensed to: University of Colorado Colorado Springs


Please enter your menu choice:
     1 - Answer questions
     2 - View Student scores
     0 - exit
Enter your answer: 1
Please enter your first name: Hard
Please enter your last name: Student
Please enter which level you would like to try (1-3)
Enter your answer: 3
Please enter the operation you would like to do:
     1 - Addition
     2 - Subtraction
     3 - Multiplication
     4 - Division
     5 - A mixture of all four
Enter your answer: 5
290 - 78 =
Enter your answer: 212
Very good!
859 * 231 =
Enter your answer: 198430
Don't give up!
(Second attempt) 859 * 231 =
Enter your answer: 198429
Keep up the good work!
977 - 163 =
Enter your answer: 814
Keep up the good work!
141 * 139 =
Enter your answer: 15600
Don't give up!
(Second attempt) 141 * 139 =
Enter your answer: 15599
No. Please try again!
6715 / 79 =
Enter your answer: 85
Excellent!
820 * 772 =
Enter your answer: 633040
```

```
Nice work!
785 * 413 =
Enter your answer: 324205
Nice work!
883 * 676 =
Enter your answer: 596908
Excellent!
975 * 426 =
Enter your answer: 415350
Very good!
1410 / 15 =
Enter your answer: 94
Nice work!
Welcome to the Computer Aided Instruction System
This software licensed to: University of Colorado Colorado Springs


Please enter your menu choice:
     1 - Answer questions
     2 - View Student scores
     0 - exit
Enter your answer: 2
Displaying scores for: Easy Student
Number correct: 10
Number wrong: 3

Displaying scores for: Hard Student
Number correct: 9
Number wrong: 3

Welcome to the Computer Aided Instruction System
This software licensed to: University of Colorado Colorado Springs


Please enter your menu choice:
     1 - Answer questions
     2 - View Student scores
     0 - exit
Enter your answer:0

F:\>
```

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

F:\>FinalProject.exe DEBUG
Tests executing...



---------------
| question.h |
---------------
test_function_generateQuestion
------------------------------
Level 1:
Question 1 Generated.
Question 2 Generated.
Question 3 Generated.
Question 4 Generated.
Question 5 Generated.
Question 6 Generated.
Question 7 Generated.
Question 8 Generated.
Question 9 Generated.
Question 10 Generated.
Level 2:
Question 1 Generated.
Question 2 Generated.
Question 3 Generated.
Question 4 Generated.
Question 5 Generated.
Question 6 Generated.
Question 7 Generated.
Question 8 Generated.
Question 9 Generated.
Question 10 Generated.
Level 3:
Question 1 Generated.
Question 2 Generated.
Question 3 Generated.
Question 4 Generated.
Question 5 Generated.
Question 6 Generated.
Question 7 Generated.
Question 8 Generated.
Question 9 Generated.
Question 10 Generated.


test_function_getAnswer
-----------------------
Level 1:5 + 1 = 6 --- Matches!
Level 2:12 + 8 = 20 --- Matches!
Level 3:491 + 127 = 618 --- Matches!
```

```
Level 1:6 - 3 = 3 --- Matches!
Level 2:33 - 18 = 15 --- Matches!
Level 3:799 - 83 = 716 --- Matches!
Level 1:8 * 6 = 48 --- Matches!
Level 2:86 * 79 = 6794 --- Matches!
Level 3:656 * 614 = 402784 --- Matches!
Level 1:36 / 4 = 9 (Remainder 0) --- Matches!
Level 2:704 / 8 = 88 (Remainder 0) --- Matches!
Level 3:1035 / 23 = 45 (Remainder 0) --- Matches!
Level 1:5 + 2 = 7 --- Matches!
Level 2:59 * 17 = 1003 --- Matches!
Level 3:581 / 7 = 83 (Remainder 0) --- Matches!



---------------
| progress.h |
---------------
Getting values for an inactive module...
Is the module active? 0
Student Name:
Number true: 0
Number false: 0

--------------------------------------------------------------------------------
Setting values and activating module...

--------------------------------------------------------------------------------
Getting values for an active module...
Is the module active? 1
Student Name: Matthew Bowker
Number true: 2
Number false: 3



---------
| ui.h |
---------
Header generation
-----------------
Welcome to the Computer Aided Instruction System
This software licensed to: University of Colorado Colorado Springs



--------------------------------------------------------------------------------
Main menu generation
--------------------
Please enter your menu choice:
     1 - Answer questions
     2 - View Student scores
     0 - exit
```

```
107
108  ------------------------------------------------------------------------------
109  Ask student name
110  ----------------
111  Please enter your first name: Matthew
112  Please enter your last name: Bowker
113  The student name entered was: Matthew Bowker
114
115  ------------------------------------------------------------------------------
116  Asking the level of the questions
117  ---------------------------------
118  Please enter which level you would like to try (1-3)
119  Enter your answer: 2
120
121  ------------------------------------------------------------------------------
122  Asking the operation
123  --------------------
124  Please enter the operation you would like to do:
125       1 - Addition
126       2 - Subtraction
127       3 - Multiplication
128       4 - Division
129       5 - A mixture of all four
130  Enter your answer: 6
131  An error has occured in the program.
132  The error condition reported: That is an invalid menu choice.
133  You can try that action again, but you may get the same result :(
134
135  Enter your answer: 4
136
137  ------------------------------------------------------------------------------
138  Asking a question the first time
139  --------------------------------
140  10 - 5 =
141
142  ------------------------------------------------------------------------------
143  Asking a question the second time
144  ---------------------------------
145  (Second attempt) 10 - 5 =
146
147  ------------------------------------------------------------------------------
148  Getting an answer
149  -----------------
150  Enter your answer: 5
151  The answer entered was: 5
152
153  ------------------------------------------------------------------------------
154  Outputting student scores
155  -------------------------
156  Displaying scores for: Charlie Test
157  Number correct: 3
158  Number wrong: 2
159
```

```
160
161   ------------------------------------------------------------------------------
162   Student answered question right
163   ------------------------------
164   Very good!
165
166   ------------------------------------------------------------------------------
167   Student answered question wrong
168   ------------------------------
169   Wrong. Try once more.
170
171   ------------------------------------------------------------------------------
172   Student's choice is invalid
173   ---------------------------
174   An error has occured in the program.
175   The error condition reported: That menu choice is not valid
176   You can try that action again, but you may get the same result :(
177
178
179   ------------------------------------------------------------------------------
180   Final Test Status (Question class): Succeeded!
181   Final Test Status (Progress class): Succeeded!
182   Final Test Status (UI class): Succeeded!
183   Press any key to continue . . .
```