Project 1: Multiplication Game

Thomas Bischoff

Original

Date: 5 / 16 / 2018

Problem: The purpose of this program is to create a functional game which asks the player a series of multiplication-based questions, in which it will cycle through difficulties depending on how well they are doing, have a chance of reusing problems that they previously answered incorrectly, and display the list of all the questions they answered correctly or incorrectly in through the course of the game.

Code 1: Multiplication 🡪 Project\_Multiplication.h

1. //######################################################################
2. // Program Header: Project\_Multiplication.h
3. // The function of Project\_Multiplication.h is to set the necessary
4. // functions and variables which will be used in testProject\_Game.cpp
5. // Author: Thomas Bischoff
6. // Date Created: 4/17/2018
7. //######################################################################
9. #include <iostream>
10. #include <cstdlib>
11. #include <ctime>
12. **using** **namespace** std;
14. // In Order to Avoid Inclusion of Multiple Definitions of MULTIPLICATION Class
15. #ifndef MULTIPLICATION
16. #define MULTIPLICATION
18. // Definitions of Class Multiplication
19. **class** Multiplication
20. {
21. **private**:
22. // The Variable to Represent the First Operand
23. **int** operand\_1;
24. // The Variable to Represent the Second Operand
25. **int** operand\_2;
26. // The Variable to Represent the Maximum Value for the Operands
27. **int** max\_limit;
28. **public**:
29. // Default Constructor
30. Multiplication();
31. // Specific Constructor
32. Multiplication(**int** new\_max\_limit);
33. // Display the Information Held by the Multiplication Object
34. **void** display() **const**;
35. // Return the Product of the Operands
36. **int** get\_correct\_answer() **const**;
37. // Display Hints for the User if they Guess Incorrectly Multiple Times
38. **void** hints() **const**;
39. // Check to see if the User Answered Correctly
40. **bool** check\_answer(**int** users\_input) **const**;
41. // Check to see if the One Multiplication Object is Greater than Another Multiplication Object
42. **bool** is\_bigger(**const** Multiplication &other) **const**;
43. // Set a New Max Limit
44. **void** set\_new\_max\_limit(**int** new\_max\_limit);
45. // Set New Random Values for the Operands
46. **void** new\_operands();
47. // Display the Problem to the User
48. **void** display\_problem();
49. // Reuse the Same Operands as Another Problem
50. **void** reuse\_operands(**const** Multiplication &other);
51. };
52. #endif

Code 1: Multiplication 🡪 Project\_Multiplication.cpp

1. //######################################################################
2. // Program Header: Project\_Multiplication.cpp
3. // The function of Project\_Multiplication.cpp is to specify the function of
4. // each function initialized in Project\_Multiplication.h
5. // Author: Thomas Bischoff
6. // Date Created: 4/17/2018
7. //######################################################################
9. #include “Project\_Multiplication.h”
11. // Initialize the Default Conditions for Multiplication
12. Multiplication::Multiplication()
13. {
14. // Set the Max Limit to be ‘10’
15. max\_limit = 10;
16. // Set the First Operand to be a Number Between ‘0’ and ‘10’
17. operand\_1 = rand()% 11;
18. // Set the Second Operand to be a Number Between ‘0’ and ‘10’
19. operand\_2 = rand()% 11;
20. }
22. // Initialize the Specific Conditions for Multiplication
23. Multiplication::Multiplication(**int** new\_max\_limit)
24. {
25. // Set the Max Limit to be the New Max Limit
26. max\_limit = new\_max\_limit;
27. // Set the First Operand to be a Number Between ‘0’ and the Max Limit
28. operand\_1 = rand()% (max\_limit + 1);
29. // Set the Second Operand to be a Number Between ‘0’ and the Max Limit
30. operand\_2 = rand()% (max\_limit + 1);
31. }
33. // Display the Infromation About the Current Object
34. **void** Multiplication::display() **const**
35. {
36. // Display the Max Limit
37. cout << “Max Limit = “ << max\_limit << endl;
38. // Display the First Operand
39. cout << “Operand 1 = “ << operand\_1 << endl;
40. // Display the Second Operand
41. cout << “Operand 2 = “ << operand\_2 << endl;
42. }
44. // Collects the Multiple of the Two Operands
45. **int** Multiplication::get\_correct\_answer() **const**
46. {
47. // Return the Product of the Two Operands
48. **return** (operand\_1 \* operand\_2);
49. }
51. // Display Hints for the User
52. **void** Multiplication::hints() **const**
53. {
54. // Display the Introduction to the Hints
55. cout << “Hints:” << endl;
56. // Check if the First or Second Operand is ‘0’
57. **if** ((operand\_1 == 0) || (operand\_2 == 0))
58. {
59. // Display that Anything Multiplied by ‘0’ is ‘0’
60. cout << “Anything Multiplied by Zero is Zero.” << endl;
61. }
62. // Otherwise
63. **else**
64. {
65. // Go Through all the Different Second Operands
66. **for** (**int** i = 0; i < operand\_2; i++)
67. {
68. // Display the Operand 1 and i with their Product
69. cout << operand\_1 << “ x “ << i << “ = “ << (operand\_1 \* i) << endl;
70. }
71. // Display the Operand 1 Times the Operand 2 is a Mystery
72. cout << operand\_1 << “ x “ << operand\_2 << “ = ???” << endl;
73. }
74. }
76. // Check to See if the User’s Input is Correct
77. **bool** Multiplication::check\_answer(**int** users\_input) **const**
78. {
79. // Return the Results of the Following
80. **return** ((\***this**).get\_correct\_answer() == users\_input);
81. }
83. // Check to See if this Multiplication Object is Larger than Another
84. **bool** Multiplication::is\_bigger(**const** Multiplication &other) **const**
85. {
86. // Return the Results of the Following
87. **return** ((\***this**).get\_correct\_answer() > other.get\_correct\_answer());
88. }
90. // Sets a New Max Limit for the Object
91. **void** Multiplication::set\_new\_max\_limit(**int** new\_max\_limit)
92. {
93. // Set the Max Limit to be Equal to the New Max Limit
94. max\_limit = new\_max\_limit;
95. }
97. // Set New Values for the Two Operands in the Object
98. **void** Multiplication::new\_operands()
99. {
100. // Set the First Operand to be a Random Number Between ‘0’ and Max Limit
101. operand\_1 = rand()% (max\_limit + 1);
102. // Set the Second Operand to be a Random Number Between ‘0’ and Max Limit
103. operand\_2 = rand()% (max\_limit + 1);
104. }
106. // Displays the Problem to the User
107. **void** Multiplication::display\_problem()
108. {
109. // Display the Problem
110. cout << “Solve for “ << operand\_1 << “ x “ << operand\_2 << “.” << endl;
111. }
113. // Reuses the Same Two Operands of Another Problem
114. **void** Multiplication::reuse\_operands(**const** Multiplication &other)
115. {
116. // Set the First Operand to be the Same as the First Operand from the Other Problem
117. operand\_1 = other.operand\_1;
118. // Set the Second Operand to be the Same as the Second Operand from the Other Problem
119. operand\_2 = other.operand\_2;
120. }

Code 2: Answers 🡪 Project\_Answer.h

1. //######################################################################
2. // Program Header: Project\_Answer.h
3. // The function of Project\_Answer.h is to set the necessary functions
4. // and variables Which Will be Used in testProject\_Game.cpp
5. // Author: Thomas Bischoff
6. // Date Created: 4/17/2018
7. //######################################################################
9. #include <iostream>
10. **using** **namespace** std;
12. // In Order to Avoid Inclusion of Multiple Definitions of ANSWER Class
13. #ifndef ANSWER
14. #define ANSWER
16. // Definitions of Class Answer
17. **class** Answer
18. {
19. **private**:
20. // The Variable to Hold the User’s Answer
21. **int** users\_answer;
22. **public**:
23. // Default Constructor
24. Answer();
25. // Allows the User to Enter in a Valid Answer
26. **void** enter\_valid\_answer();
27. // Returns the User’s Answer
28. **int** get\_answer();
29. };
30. #endif

Code 2: Answers 🡪 Project\_Answer.cpp

1. //######################################################################
2. // Program Header: Project\_Answer.cpp
3. // The function of Project\_Answer.cpp is to specify the functionality of
4. // each function initialized in Project\_Answer.h
5. // Author: Thomas Bischoff
6. // Date Created: 4/18/2018
7. //######################################################################
9. #include “Project\_Answer.h”
11. // Initialize the Default Conditions for Answer
12. Answer::Answer()
13. {
14. // Initialize the User’s Answer to Start Out as ‘0’
15. users\_answer = 0;
16. }
18. // Allows the User to Enter in a Valid Answer
19. **void** Answer::enter\_valid\_answer()
20. {
21. // Initilaize a Variable to Hold the User’s Input
22. **int** users\_input;
23. // Ask the User to Input the Answer
24. cout << “Please Enter in the Answer (Enter -1 if You want to End Game): “;
25. // Take the User’s Input and Have it Held by the Variable
26. cin >> users\_input;
27. // Check to See that the Answer is Less than ‘0’
28. **while** ((users\_input < 0) || (users\_input > 225))
29. {
30. // Check if the User’s Input is ‘-1’
31. **if** (users\_input == -1)
32. {
33. // Display that the Game will Now End
34. cout << “Exitiing the Game.” << endl;
35. // Break the While Loop
36. **break**;
37. }
38. // Otherwise
39. **else**
40. {
41. // Display a Message Indicating that the Value Entered was not Valid
42. cout << “Sorry, but the Value ‘” << users\_input << “’ is not a Valid Input.” << endl;
43. // Ask the User to Input the Answer
44. cout << “Please Enter in the Answer: “;
45. // Take the User’s Input and Have it Held by the Variable
46. cin >> users\_input;
47. }
48. }
49. // Set User’s Answer to be User’s Input
50. users\_answer = users\_input;
51. }
53. // Returns the User’s Answer
54. **int** Answer::get\_answer()
55. {
56. // Return User’s Answer
57. **return** users\_answer;
58. }

Code 3: Problems 🡪 Project\_Problem.h

1. //######################################################################
2. // Program Header: Project\_Problem.h
3. // The function of Project\_Problem.h is to set the necessary functions
4. // and variables which will be used in testProject\_Game.cpp.
5. // Author: Thomas Bischoff
6. // Date Created: 4/18/2018
7. //######################################################################
9. #include <iostream>
10. #include “Project\_Multiplication.h”
11. #include “Project\_Answer.h”
12. **using** **namespace** std;
14. // In Order to Avoid Inclusion of Multiple Definitions of PROBLEM Class
15. #ifndef PROBLEM
16. #define PROBLEM
18. // Definitions of Class Multiplication
19. **class** Problem
20. {
21. **private**:
22. // Initialize the Question
23. Multiplication question;
24. // Initialize the Answer
25. Answer answer;
26. // The Variable to Hold the Max Limit of the Operands
27. **int** max\_limit\_operand;
28. // The Variable to Hold the Max Amount of Attempts the User Gets
29. **int** max\_tries;
30. // The Variable to Hold the Current Number of Incorrect Guesses for the Question
31. **int** number\_of\_incorrect;
32. **public**:
33. // Default Constructor
34. Problem();
35. // Specific Constructor
36. Problem(**int** new\_max\_limit\_operand, **int** new\_max\_tries);
37. // Creates a New Problem for the User
38. **void** set\_new\_problem();
39. // Allows the User to Solve the Problem
40. **int** solve\_problem();
41. // Return the Number of Incorrect Guesses Made on the Problem
42. **int** get\_number\_of\_incorrect();
43. // Changes the Number of Tries
44. **void** change\_number\_of\_tries(**int** new\_tries);
45. // Changes the Max Limit Operand
46. **void** change\_max\_operand(**int** new\_operand);
47. // Redo a Problem
48. **void** redo\_problem(**const** Problem &other);
49. // Display the Question and the Number of Incorrect Guesses
50. **void** display(Problem &other);
51. // Display the Question
52. **void** display\_question();
53. };
54. #endif

Code 3: Problems 🡪 Project\_Problem.cpp

1. //######################################################################
2. // Program Header: Project\_Problem.cpp
3. // The function of Project\_Problem.cpp is specify the function of each
4. // function initialized in Project\_Problem.h
5. // Author: Thomas Bischoff
6. // Date Created: 4/18/2018
7. //######################################################################
9. #include “Project\_Problem.h”
11. // Initialize the Default Problem
12. Problem::Problem()
13. {
14. // Set the Max Limit Operand to be ‘10’
15. max\_limit\_operand = 10;
16. // Set the Max Number of Tries to be ‘3’
17. max\_tries = 3;
18. // Set the Number of Incorrect Guesses to be ‘0’
19. number\_of\_incorrect = 0;
20. // Randomly Choose New Operands for the Question
21. question.new\_operands();
22. }
24. // Initialize the Specific Problem
25. Problem::Problem(**int** new\_max\_limit\_operand, **int** new\_max\_tries)
26. {
27. // Set the Max Limit Operand to be the Same as New Max Limit Operand
28. max\_limit\_operand = new\_max\_limit\_operand;
29. // Set the Max Number of Tries to be the Same as New Max Tries
30. max\_tries = new\_max\_tries;
31. // Set the Number of Incorrect Guesses to be ‘0’
32. number\_of\_incorrect = 0;
33. // Set the New Max Limit for the Question to be the Max Limit Operand
34. question.set\_new\_max\_limit(max\_limit\_operand);
35. // Randomly Choose New Operands for the Question
36. question.new\_operands();
37. }
39. // Sets Up a New Problem for the User
40. **void** Problem::set\_new\_problem()
41. {
42. // Reset the Number of Incorrect Guesses to be ‘0’
43. number\_of\_incorrect = 0;
44. // Randomly Choose New Operands for the Next Question
45. question.new\_operands();
46. }
48. // Have the User Try to Solve the Problem and Return the Number of Incorrect Guesses that they Took
49. **int** Problem::solve\_problem()
50. {
51. // Display the Question to the User
52. question.display\_problem();
53. // Generate a WHILE Loop that Will Check if the Number of Incorrect Guesses is Less than the Max Tries
54. **while** (number\_of\_incorrect < max\_tries)
55. {
56. // Check to See if the Number of Incorrect Guesses is One Less than the Max Number of Incorrect Guesses
57. **if** ((number\_of\_incorrect == (max\_tries – 1)) && (number\_of\_incorrect != 0))
58. {
59. // Display the Hint to the User
60. question.hints();
61. }
62. // Have the User Input what they Believe to be the Answer
63. answer.enter\_valid\_answer();
64. // Check to See if the User’s Answer is the Same as the is the Correct Answer
65. **if** (question.check\_answer(answer.get\_answer()))
66. {
67. // Display that the User is Correct
68. cout << “You are Correct!!!” << endl;
69. // Return the Current Number of Incorrect Guesses
70. **return** number\_of\_incorrect;
71. }
72. // Check to See if the Player Entered in ‘-1’
73. **else** **if** (answer.get\_answer() == -1)
74. {
75. // Return ‘-1’
76. **return** -1;
77. }
78. // Otherwise
79. **else**
80. {
81. // Increase the Number of Incorrect Guesses by ‘1’
82. number\_of\_incorrect += 1;
83. // Display that the User is Incorrect
84. cout << “Sorry, but ‘” << answer.get\_answer() << “’ is Not the Answer.” << endl;
85. // Display the Number of Tries that the User has Left
86. cout << “You Currently Have ‘” << (max\_tries – number\_of\_incorrect) << “’ Tries Remaining.” << endl;
87. }
88. }
89. // Display that the User did not get the Question Correct
90. cout << “Sorry, You are Completely Out of Guesses.” << endl;
91. // Display the Correct Answer to the User
92. cout << “The Correct Answer is ‘” << question.get\_correct\_answer() << “’.” << endl;
93. // Return the Number of Incorrect Guesses
94. **return** number\_of\_incorrect;
95. }
97. // Return the Number of Incorrect Guesses
98. **int** Problem::get\_number\_of\_incorrect()
99. {
100. // Return Number of Incorrect
101. **return** number\_of\_incorrect;
102. }
104. // Changes the Number of Tries that the User Has
105. **void** Problem::change\_number\_of\_tries(**int** new\_tries)
106. {
107. // Sets Max Tries to be the New Number of Tries
108. max\_tries = new\_tries;
109. }
111. // Changes the Max Operand
112. **void** Problem::change\_max\_operand(**int** new\_operand)
113. {
114. // Sets Max Limit Operand to be the New Operand
115. max\_limit\_operand = new\_operand;
116. // Set the Question to Have the New Operand
117. question.set\_new\_max\_limit(max\_limit\_operand);
118. }
120. // Allows the User to Redo a Problem they Got Wrong
121. **void** Problem::redo\_problem(**const** Problem &other)
122. {
123. // Set the Number of Incorrect Guesses to be ‘0’
124. number\_of\_incorrect = 0;
125. // Set the Question to be the Same
126. question.reuse\_operands(other.question);
127. }
129. // Display the Question and Number of Incorrect Guesses
130. **void** Problem::display(Problem &other)
131. {
132. // Display the Question
133. other.display\_question();
134. // Display the Number of Incorrect Guesses
135. cout << “Number of Incorrect Guesses = “ << other.number\_of\_incorrect << endl;
136. }
138. // Display the Problem
139. **void** Problem::display\_question()
140. {
141. // Display the Question
142. question.display\_problem();
143. }

Code 4: Game 🡪 Project\_Game.h

1. //######################################################################
2. // Program Header: Project\_Game.h
3. // The function of Project\_Game.h is to set the necessary functions and
4. // variables which will be used in testProject\_Game.cpp
5. // Author: Thomas Bischoff
6. // Date Created: 4/20/2018
7. //######################################################################
9. #include <iostream>
10. #include <cctype>
11. #include <list>
12. #include <cstdlib>
13. #include <ctime>
14. #include "Project\_Problem.h"
15. **using** **namespace** std;
17. // In Order to Avoid Inclusion of Multiple Definitions of GAME Class
18. #ifndef GAME
19. #define GAME
21. // Definition of Class Game
22. **class** Game
23. {
24. **private**:
25. // The Problems in the Game
26. Problem problem;
27. // The Variable to Represent the Current Difficulty
28. **int** difficulty;
29. // The Variable to Represent the Total Number of Problems
30. **int** number\_of\_problems;
31. // List of the Problems that the Player Got Correct on their First Attempt
32. std::list<Problem> first\_attempt\_list;
33. // List of the Problems that the Player Got Incorrect and Number of Guesses Made
34. std::list<Problem> incorrect\_list;
35. // The Variable to Represent if the User Wants to Play Again
36. **bool** play;
37. **public**:
38. // Default Constructor
39. Game();
40. // Specific Constructor
41. Game(**int** new\_max\_limit\_operand, **int** new\_max\_tries);
42. // Allows User to Play the Game
43. **void** play\_game();
44. // Creates a New Problem
45. **void** generate\_new\_sentence();
46. // Updates the Max Operand Limit
47. **void** update\_limit(**int** \*array, **int** position);
48. // Updates the Player's Stats
49. **void** update\_stats();
50. // Display the User's Stats
51. **void** print\_stats(**int** hardness);
52. // Updates the Play Variable
53. **void** play\_again();
54. // Set up a New Game
55. **void** set\_new\_game();
56. // Displays the Information Within the Lists
57. **void** display\_lists();
58. };
59. #endif

Code 4: Game 🡪 Project\_Game.cpp

1. //######################################################################
2. // Program Header: Project\_Game.cpp
3. // The function of Project\_Game.cpp is specify the function of each
4. // funtion initialized in Project\_Game.h
5. // Author: Thomas Bischoff
6. // Date Created: 4/20/2018
7. //######################################################################
9. #include "Project\_Game.h"
11. // Initialize the Default Game
12. Game::Game()
13. {
14. // Display the Intro to the Difficulty Selection
15. cout << "Difficulty Selection:" << endl;
16. // Display the Information for the Easy Difficulty
17. cout << "1. Easy (Number of Questions = 10, Max Operand = 5, Tries = 5)" << endl;
18. // Display the Information for the Medium Difficulty
19. cout << "2. Medium (Number of Questions = 15, Max Operand = 10, Tries = 3)" << endl;
20. // Display the Information for the Hard Difficulty
21. cout << "3. Hard (Number of Questions = 20, Max Operand = 15, Tries = 1)" << endl;
22. // Initialize a Statement as True
23. **bool** run\_difficulty = **true**;
24. // Generate a WHILE Loop
25. **while** (run\_difficulty)
26. {
27. // Ask the User to Input the Difficulty they Want
28. cout << "Please Choose a Difficulty by Entering in its Corresponding Number: ";
29. // Take the User's Input and Set it to be Held by Difficulty
30. cin >> difficulty;
31. // Check to See if the Number the User Entered is Valid
32. **if** ((difficulty >= 1) && (difficulty <= 3))
33. {
34. // Change the Statement to be False
35. run\_difficulty = **false**;
36. }
37. // Otherwise
38. **else**
39. {
40. // Display that the User's Input is Not Valid
41. cout << "Sorry, but the Input '" << difficulty << "' is Not a Valid Difficulty." << endl;
42. }
43. }
44. // Sets the Number of Questions
45. number\_of\_problems = 5 \* (difficulty + 1);
46. // Initialize a Value for the Max Operand
47. **int** max\_operand = 5 \* difficulty;
48. // Initialize a Value for the Number of Tries
49. **int** number\_of\_tries = 7 - (2 \* difficulty);
50. // Set the Problem with the New Max Operand
51. problem.change\_max\_operand(max\_operand);
52. // Set the Problem with the New Max Tries
53. problem.change\_number\_of\_tries(number\_of\_tries);
54. // Set Play to be True
55. play = **true**;
56. }
58. // Initialize the Specific Game
59. Game::Game(**int** new\_max\_operand, **int** new\_max\_tries)
60. {
61. // Display the Intro to the Difficulty Selection
62. cout << "Difficulty Selection:" << endl;
63. // Display the Information for the Easy Difficulty
64. cout << "1. Easy (Number of Questions = 10)" << endl;
65. // Display the Information for the Medium Difficulty
66. cout << "2. Medium (Number of Questions = 15)" << endl;
67. // Display the Information for the Hard Difficulty
68. cout << "3. Hard (Number of Questions = 20)" << endl;
69. // Initialize a Statement as True
70. **bool** run\_difficulty = **true**;
71. // Generate a WHILE Loop
72. **while** (run\_difficulty)
73. {
74. // Ask the User to Input the Difficulty they Want
75. cout << "Please Choose a Difficulty by Entering in its Corresponding Number: ";
76. // Take the User's Input and Set it to be Held by Difficulty
77. cin >> difficulty;
78. // Check to See if the Number the User Entered is Valid
79. **if** ((difficulty >= 1) && (difficulty <= 3))
80. {
81. // Change the Statement to be False
82. run\_difficulty = **false**;
83. }
84. // Otherwise
85. **else**
86. {
87. // Display that the User's Input is Not Valid
88. cout << "Sorry, but the Input '" << difficulty << "' is Not a Valid Difficulty." << endl;
89. }
90. }
91. // Sets the Number of Questions
92. number\_of\_problems = 5 \* (difficulty + 1);
93. // Set the Problem with the New Max Operand
94. problem.change\_max\_operand(new\_max\_operand);
95. // Set the Problem with the New Max Number of Tries
96. problem.change\_number\_of\_tries(new\_max\_tries);
97. // Sets Play to be True
98. play = **true**;
99. }
101. **void** Game::play\_game()
102. {
103. // Generate a WHILE Loop to Check if Play is True
104. **while** (play)
105. {
106. // Initialize a Value for the Starting Difficulty
107. **int** starting\_difficulty = difficulty;
108. // Create an Array to Hold the Number of Guesses the User Makes Per Question
109. **int** guesses\_array[number\_of\_problems];
110. // Generate a FOR Loop which will Last for as Many Questions are in the Game
111. **for** (**int** i = 0; i < number\_of\_problems; i++)
112. {
113. // Mark off the Question
114. cout << "---------------------------------------------------" << endl;
115. // Display the Question Number
116. cout << "Question " << (i + 1) << ":" << endl;
117. // Run Generate New Sentence
118. (\***this**).generate\_new\_sentence();
119. // Have the Player Answer the Problem
120. **int** number\_of\_incorrect = problem.solve\_problem();
121. // Check if the Number of Incorrect is '-1'
122. **if** (number\_of\_incorrect == -1)
123. {
124. // Break out of the FOR Loop
125. **break**;
126. }
127. // Add the Number of Incorrect to the Number of Guesses
128. guesses\_array[i] = number\_of\_incorrect;
129. // Check to See if i is Greater than or Equal to 2
130. **if** (i >= 2)
131. {
132. // Run Update Limit
133. (\***this**).update\_limit(guesses\_array, i);
134. }
135. // Update the Stats
136. (\***this**).update\_stats();
137. }
138. // Run Print Stats
139. (\***this**).print\_stats(starting\_difficulty);
140. // Run Play Again
141. (\***this**).play\_again();
142. }
143. }
145. // Generates a New Problem for the Player
146. **void** Game::generate\_new\_sentence()
147. {
148. // Generate a Random Number Between '1' and '100'
149. **int** random\_number = rand()% 100 + 1;
150. // Check if the Number is Less than '75'
151. **if** (random\_number < 75)
152. {
153. // Set a New Problem
154. problem.set\_new\_problem();
155. }
156. // Otherwise
157. **else**
158. {
159. // Check if the List of Incorrect Questions is Empty
160. **if** (incorrect\_list.empty())
161. {
162. // Set a New Problem
163. problem.set\_new\_problem();
164. }
165. // Otherwise
166. **else**
167. {
168. // Display a Message Saying that an Old Question is Being Reused
169. cout << "A Question You Previously Answered Wrong is Going to be Asked." << endl;
170. // Set the Problem to be the Same as the First Problem in the Incorrect List
171. problem.redo\_problem(\*incorrect\_list.begin());
172. // Remove the Question from the List
173. incorrect\_list.pop\_front();
174. }
175. }
176. }
178. // Updates the Max Limit of the Problems Depending on How Well the Player Does
179. **void** Game::update\_limit(**int** \*array, **int** position)
180. {
181. // Check to See if the Player's Difficulty is '1'
182. **if** (difficulty == 1)
183. {
184. // Check if the the Last Three Questions Have Been Answered Correctly on the First Try
185. **if** (array[position] == 0 && array[position - 1] == 0 && array[position - 2] == 0)
186. {
187. // Tell the User that the Diifculty has Been Raised to Medium
188. cout << "Question Difficulty has Been Raised to Medium." << endl;
189. // Set the Difficult to be '2'
190. difficulty = 2;
191. // Set the Problem to Have a New Max Operand
192. problem.change\_max\_operand(5 \* difficulty);
193. }
194. }
195. // Check to See if the Player's Difficulty is '2'
196. **else** **if** (difficulty == 2)
197. {
198. // Check if the the Last Three Questions Have Been Answered Correctly on the First Try
199. **if** (array[position] == 0 && array[position - 1] == 0 && array[position - 2] == 0)
200. {
201. // Tell the User that the Diifculty has Been Raised to Hard
202. cout << "Question Difficulty has Been Raised to Hard." << endl;
203. // Set the Difficult to be '3'
204. difficulty = 3;
205. // Set the Problem to Have a New Max Operand
206. problem.change\_max\_operand(5 \* difficulty);
207. }
208. // Check if the Last Three Questions Have Been Answered Incorrectly
209. **if** (array[position] == (7 - (2 \* difficulty)) && array[position - 1] == (7 - (2 \* difficulty)) && array[position - 2] == (7 - (2 \* difficulty)))
210. {
211. // Tell the the User that the Difficulty has Been Reduced to Easy
212. cout << "Question Difficult has Been Lowered to Easy." << endl;
213. // Set the Difficult to be '1'
214. difficulty = 1;
215. // Set the Problem to Have a New Max Operand
216. problem.change\_max\_operand(5 \* difficulty);
217. }
218. }
219. // Check to See if the Player's Difficulty is '3'
220. **else** **if** (difficulty == 3)
221. {
222. // Check if the Last Three Questions Have Been Answered Incorrectly
223. **if** (array[position] == (7 - (2 \* difficulty)) && array[position - 1] == (7 - (2 \* difficulty)) && array[position - 2] == (7 - (2 \* difficulty)))
224. {
225. // Tell the the User that the Difficulty has Been Reduced to Medium
226. cout << "Question Difficult has Been Lowered to Medium." << endl;
227. // Set the Difficult to be '2'
228. difficulty = 2;
229. // Set the Problem to Have a New Max Operand
230. problem.change\_max\_operand(5 \* difficulty);
231. }
232. }
233. }
235. // Update the Players Stats for the Game
236. **void** Game::update\_stats()
237. {
238. // Check if the Number of Incorrect Guesses is '0'
239. **if** (problem.get\_number\_of\_incorrect() == 0)
240. {
241. // Add the Problem to the Correct List
242. first\_attempt\_list.push\_back(problem);
243. }
244. // Otherwise
245. **else**
246. {
247. // Add the Problem to the Incorrect List
248. incorrect\_list.push\_back(problem);
249. }
250. }
252. // Display Player's Stats for the Game
253. **void** Game::print\_stats(**int** hardness)
254. {
255. // Display the Title Player Stats
256. cout << "Game Stats:" << endl;
257. // Check if the Hardness is Different than the Starting Difficulty
258. **if** (hardness != difficulty)
259. {
260. // Display the Starting Difficulty
261. cout << "Starting Difficulty = " << hardness << endl;
262. // Display the Starting Max Operand
263. cout << "Starting Max Operand = " << (5 \* hardness) << endl;
264. // Display the Ending Difficulty
265. cout << "Ending Difficulty = " << difficulty << endl;
266. // Display the Ending Max Operand
267. cout << "Ending Max Operand = " << (5 \* difficulty) << endl;
268. // Display the Number of Questions
269. cout << "Number of Questions = " << (5 \* (hardness + 1)) << endl;
270. // Display the Max Number of Tries
271. cout << "Number of Tries = " << (7 - (2 \* hardness)) << endl;
272. // Display the Information from the Lists
273. (\***this**).display\_lists();
274. }
275. // Otherwise
276. **else**
277. {
278. // Display the Difficulty
279. cout << "Difficulty = " << difficulty << endl;
280. // Display the Max Operand
281. cout << "Max Operand = " << (5 \* difficulty) << endl;
282. // Display the Number of Questions
283. cout << "Number of Questions = " << (5 \* (difficulty + 1)) << endl;
284. // Display the Number of Tries
285. cout << "Number of Tries = " << (7 - (2 \* difficulty)) << endl;
286. // Display the Information from the Lists
287. (\***this**).display\_lists();
288. }
289. }

292. // Allows the User to Play the Game Again
293. **void** Game::play\_again()
294. {
295. // Set a Statement as True
296. **bool** check = **true**;
297. // Generate a WHILE Loop
298. **while** (check)
299. {
300. // Initialize a Variable to Hold the User's Response
301. **char** users\_response;
302. // Ask if the Player if they Want to Play Again
303. cout << "Would You Like to Play Again (Y/N): ";
304. // Take the User's Response
305. cin >> users\_response;
306. // Convert the User's Response to Uppercase
307. users\_response = toupper(users\_response);
308. // Check if the User Entered 'Y'
309. **if** (users\_response == 'Y')
310. {
311. // Set Play to be True
312. play = **true**;
313. // Set Check to be False
314. check = **false**;
315. }
316. // Check if the User Entered 'N'
317. **else** **if** (users\_response == 'N')
318. {
319. // Set Play to be False
320. play = **false**;
321. // Set Check to be False
322. check = **false**;
323. }
324. // Otherwise
325. **else**
326. {
327. // Display that their Input is Not Recognized
328. cout << "Sorry, but the Input You Have Entered is Not Recognized. Please Try Again." << endl;
329. }
330. }
331. // Check if the User Wants to Play Again
332. **if** (play)
333. {
334. // Set Up the New Game
335. (\***this**).set\_new\_game();
336. }
337. // Otherwise
338. **else**
339. {
340. // Display a Goodbye Message
341. cout << "Goodbye." << endl;
342. }
343. }
345. // Sets Up a New Game for the User
346. **void** Game::set\_new\_game()
347. {
348. // Display the Intro to the Difficulty Selection
349. cout << "Difficulty Selection:" << endl;
350. // Display the Information for the Easy Difficulty
351. cout << "1. Easy (Number of Questions = 10, Max Operand = 5, Tries = 5)" << endl;
352. // Display the Information for the Medium Difficulty
353. cout << "2. Medium (Number of Questions = 15, Max Operand = 10, Tries = 3)" << endl;
354. // Display the Information for the Hard Difficulty
355. cout << "3. Hard (Number of Questions = 20, Max Operand = 15, Tries = 1)" << endl;
356. // Initialize a Statement as True
357. **bool** run\_difficulty = **true**;
358. // Generate a WHILE Loop
359. **while** (run\_difficulty)
360. {
361. // Ask the User to Input the Difficulty they Want
362. cout << "Please Choose a Difficulty by Entering in its Corresponding Number: ";
363. // Take the User's Input and Set it to be Held by Difficulty
364. cin >> difficulty;
365. // Check to See if the Number the User Entered is Valid
366. **if** ((difficulty >= 1) && (difficulty <= 3))
367. {
368. // Change the Statement to be False
369. run\_difficulty = **false**;
370. }
371. // Otherwise
372. **else**
373. {
374. // Display that the User's Input is Not Valid
375. cout << "Sorry, but the Input '" << difficulty << "' is Not a Valid Difficulty." << endl;
376. }
377. }
378. // Sets the Number of Questions
379. number\_of\_problems = 5 \* (difficulty + 1);
380. // Empty the First Attempt List
381. first\_attempt\_list.clear();
382. // Empty the Incorrect List
383. incorrect\_list.clear();
384. // Initialize a Value for the Max Operand
385. **int** max\_operand = 5 \* difficulty;
386. // Initialize a Value for the Number of Tries
387. **int** number\_of\_tries = 7 - (2 \* difficulty);
388. // Set the Problem with the New Max Operand
389. problem.change\_max\_operand(max\_operand);
390. // Set the Problem with the New Max Tries
391. problem.change\_number\_of\_tries(number\_of\_tries);
392. }
394. // Displays the Information within the Lists
395. **void** Game::display\_lists()
396. {
397. // Display the Title for the Questions Answered Corrrectly
398. cout << "Answered Correctly:" << endl;
399. // Check if the List Holding the Number of Questions Answered Correctly is Empty
400. **if** (first\_attempt\_list.empty())
401. {
402. // Display that No Answers were Correct on the First Guess
403. cout << "No Questions were Answered Correctly on the First Attempt." << endl;
404. }
405. // Otherwise
406. **else**
407. {
408. // Initialize a Varible for the Size of the List of Correct
409. **int** size\_of\_correct = first\_attempt\_list.size();
410. // Go Through Each Element in the List
411. **for** (**int** i = 0; i < size\_of\_correct; i++)
412. {
413. // Display the Information About the Problem in the List
414. problem.display(\*first\_attempt\_list.begin());
415. // Remove the Item from the List
416. first\_attempt\_list.pop\_front();
417. }
418. }
419. // Display the Title for the Questions Not Answered Correctly on the First Attempt
420. cout << "Not Answered Correctly:" << endl;
421. // Check if the List Holding the Number of Questions Answered Incorrectly is Empty
422. **if** (incorrect\_list.empty())
423. {
424. // Display that No Answers were Answered Incorrectly
425. cout << "No Questions were Answered Incorrectly." << endl;
426. }
427. // Otherwise
428. **else**
429. {
430. // Initialize a Varible for the Size of the List of Incorrect
431. **int** size\_of\_incorrect = incorrect\_list.size();
432. // Go Through the Elements in the List
433. **for** (**int** j = 0; j < size\_of\_incorrect; j++)
434. {
435. // Display the Information About the Problem in the List
436. problem.display(\*incorrect\_list.begin());
437. // Remove the Item from the List
438. incorrect\_list.pop\_front();
439. }
440. }
441. }

Code 5: Test 🡪 testProject\_Game.cpp

1. //######################################################################
2. // Program Header: testProject\_Game.cpp
3. // The Function of this Program is to Test all the Elements of the Game
4. // Author: Thomas Bischoff
5. // Date Created: 5/2/2018
6. //######################################################################
8. #include <iostream>
9. #include "Project\_Game.h"
10. **using** **namespace** std;
12. **int** main()
13. {
14. // Set the Time
15. srand(time(0));
16. // Initialize a Default Game
17. Game game\_1;
18. // Play the Game
19. game\_1.play\_game();
20. }

Sample Output:

1. Difficulty Selection:
2. 1. Easy (Number of Questions = 10, Max Operand = 5, Tries = 5)
3. 2. Medium (Number of Questions = 15, Max Operand = 10, Tries = 3)
4. 3. Hard (Number of Questions = 20, Max Operand = 15, Tries = 1)
5. Please Choose a Difficulty by Entering in its Corresponding Number: 1
6. ---------------------------------------------------
7. Question 1:
8. Solve for 5 x 2.
9. Please Enter in the Answer (Enter -1 if You want to End Game): 10
10. You are Correct!!!
11. ---------------------------------------------------
12. Question 2:
13. Solve for 0 x 3.
14. Please Enter in the Answer (Enter -1 if You want to End Game): 0
15. You are Correct!!!
16. ---------------------------------------------------
17. Question 3:
18. Solve for 2 x 0.
19. Please Enter in the Answer (Enter -1 if You want to End Game): 0
20. You are Correct!!!
21. Question Difficulty has Been Raised to Medium.
22. ---------------------------------------------------
23. Question 4:
24. Solve for 6 x 3.
25. Please Enter in the Answer (Enter -1 if You want to End Game): 18
26. You are Correct!!!
27. Question Difficulty has Been Raised to Hard.
28. ---------------------------------------------------
29. Question 5:
30. Solve for 3 x 9.
31. Please Enter in the Answer (Enter -1 if You want to End Game): 27
32. You are Correct!!!
33. ---------------------------------------------------
34. Question 6:
35. Solve for 2 x 7.
36. Please Enter in the Answer (Enter -1 if You want to End Game): 14
37. You are Correct!!!
38. ---------------------------------------------------
39. Question 7:
40. Solve for 1 x 11.
41. Please Enter in the Answer (Enter -1 if You want to End Game): 11
42. You are Correct!!!
43. ---------------------------------------------------
44. Question 8:
45. Solve for 7 x 3.
46. Please Enter in the Answer (Enter -1 if You want to End Game): 21
47. You are Correct!!!
48. ---------------------------------------------------
49. Question 9:
50. Solve for 9 x 0.
51. Please Enter in the Answer (Enter -1 if You want to End Game): 0
52. You are Correct!!!
53. ---------------------------------------------------
54. Question 10:
55. Solve for 10 x 10.
56. Please Enter in the Answer (Enter -1 if You want to End Game): 10
57. Sorry, but '10' is Not the Answer.
58. You Currently Have '4' Tries Remaining.
59. Please Enter in the Answer (Enter -1 if You want to End Game): 100
60. You are Correct!!!
61. Game Stats:
62. Starting Difficulty = 1
63. Starting Max Operand = 5
64. Ending Difficulty = 3
65. Ending Max Operand = 15
66. Number of Questions = 10
67. Number of Tries = 5
68. Answered Correctly:
69. Solve for 5 x 2.
70. Number of Incorrect Guesses = 0
71. Solve for 0 x 3.
72. Number of Incorrect Guesses = 0
73. Solve for 2 x 0.
74. Number of Incorrect Guesses = 0
75. Solve for 6 x 3.
76. Number of Incorrect Guesses = 0
77. Solve for 3 x 9.
78. Number of Incorrect Guesses = 0
79. Solve for 2 x 7.
80. Number of Incorrect Guesses = 0
81. Solve for 1 x 11.
82. Number of Incorrect Guesses = 0
83. Solve for 7 x 3.
84. Number of Incorrect Guesses = 0
85. Solve for 9 x 0.
86. Number of Incorrect Guesses = 0
87. Not Answered Correctly:
88. Solve for 10 x 10.
89. Number of Incorrect Guesses = 1
90. Would You Like to Play Again (Y/N): y
91. Difficulty Selection:
92. 1. Easy (Number of Questions = 10, Max Operand = 5, Tries = 5)
93. 2. Medium (Number of Questions = 15, Max Operand = 10, Tries = 3)
94. 3. Hard (Number of Questions = 20, Max Operand = 15, Tries = 1)
95. Please Choose a Difficulty by Entering in its Corresponding Number: 2
96. ---------------------------------------------------
97. Question 1:
98. Solve for 4 x 7.
99. Please Enter in the Answer (Enter -1 if You want to End Game): 13112
100. Sorry, but the Value '13112' is not a Valid Input.
101. Please Enter in the Answer: -12
102. Sorry, but the Value '-12' is not a Valid Input.
103. Please Enter in the Answer: 28
104. You are Correct!!!
105. ---------------------------------------------------
106. Question 2:
107. Solve for 7 x 0.
108. Please Enter in the Answer (Enter -1 if You want to End Game): 0
109. You are Correct!!!
110. ---------------------------------------------------
111. Question 3:
112. Solve for 1 x 7.
113. Please Enter in the Answer (Enter -1 if You want to End Game): 7
114. You are Correct!!!
115. Question Difficulty has Been Raised to Hard.
116. ---------------------------------------------------
117. Question 4:
118. Solve for 5 x 13.
119. Please Enter in the Answer (Enter -1 if You want to End Game): 65
120. You are Correct!!!
121. ---------------------------------------------------
122. Question 5:
123. Solve for 11 x 0.
124. Please Enter in the Answer (Enter -1 if You want to End Game): 0
125. You are Correct!!!
126. ---------------------------------------------------
127. Question 6:
128. Solve for 1 x 8.
129. Please Enter in the Answer (Enter -1 if You want to End Game): 8
130. You are Correct!!!
131. ---------------------------------------------------
132. Question 7:
133. Solve for 2 x 0.
134. Please Enter in the Answer (Enter -1 if You want to End Game): 0
135. You are Correct!!!
136. ---------------------------------------------------
137. Question 8:
138. Solve for 10 x 13.
139. Please Enter in the Answer (Enter -1 if You want to End Game): 130
140. You are Correct!!!
141. ---------------------------------------------------
142. Question 9:
143. Solve for 9 x 9.
144. Please Enter in the Answer (Enter -1 if You want to End Game): 81
145. You are Correct!!!
146. ---------------------------------------------------
147. Question 10:
148. Solve for 0 x 0.
149. Please Enter in the Answer (Enter -1 if You want to End Game): -1
150. Exitiing the Game.
151. Game Stats:
152. Starting Difficulty = 2
153. Starting Max Operand = 10
154. Ending Difficulty = 3
155. Ending Max Operand = 15
156. Number of Questions = 15
157. Number of Tries = 3
158. Answered Correctly:
159. Solve for 4 x 7.
160. Number of Incorrect Guesses = 0
161. Solve for 7 x 0.
162. Number of Incorrect Guesses = 0
163. Solve for 1 x 7.
164. Number of Incorrect Guesses = 0
165. Solve for 5 x 13.
166. Number of Incorrect Guesses = 0
167. Solve for 11 x 0.
168. Number of Incorrect Guesses = 0
169. Solve for 1 x 8.
170. Number of Incorrect Guesses = 0
171. Solve for 2 x 0.
172. Number of Incorrect Guesses = 0
173. Solve for 10 x 13.
174. Number of Incorrect Guesses = 0
175. Solve for 9 x 9.
176. Number of Incorrect Guesses = 0
177. Not Answered Correctly:
178. No Questions were Answered Incorrectly.
179. Would You Like to Play Again (Y/N): y
180. Difficulty Selection:
181. 1. Easy (Number of Questions = 10, Max Operand = 5, Tries = 5)
182. 2. Medium (Number of Questions = 15, Max Operand = 10, Tries = 3)
183. 3. Hard (Number of Questions = 20, Max Operand = 15, Tries = 1)
184. Please Choose a Difficulty by Entering in its Corresponding Number: 3
185. ---------------------------------------------------
186. Question 1:
187. Solve for 15 x 14.
188. Please Enter in the Answer (Enter -1 if You want to End Game): 160
189. Sorry, but '160' is Not the Answer.
190. You Currently Have '0' Tries Remaining.
191. Sorry, You are Completely Out of Guesses.
192. The Correct Answer is '210'.
193. ---------------------------------------------------
194. Question 2:
195. Solve for 12 x 12.
196. Please Enter in the Answer (Enter -1 if You want to End Game): 144
197. You are Correct!!!
198. ---------------------------------------------------
199. Question 3:
200. Solve for 7 x 2.
201. Please Enter in the Answer (Enter -1 if You want to End Game): 14
202. You are Correct!!!
203. ---------------------------------------------------
204. Question 4:
205. Solve for 6 x 2.
206. Please Enter in the Answer (Enter -1 if You want to End Game): 12
207. You are Correct!!!
208. ---------------------------------------------------
209. Question 5:
210. A Question You Previously Answered Wrong is Going to be Asked.
211. Solve for 15 x 14.
212. Please Enter in the Answer (Enter -1 if You want to End Game): -1
213. Exitiing the Game.
214. Game Stats:
215. Difficulty = 3
216. Max Operand = 15
217. Number of Questions = 20
218. Number of Tries = 1
219. Answered Correctly:
220. Solve for 12 x 12.
221. Number of Incorrect Guesses = 0
222. Solve for 7 x 2.
223. Number of Incorrect Guesses = 0
224. Solve for 6 x 2.
225. Number of Incorrect Guesses = 0
226. Not Answered Correctly:
227. No Questions were Answered Incorrectly.
228. Would You Like to Play Again (Y/N): n
229. Goodbye.