```
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     }
```