```
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     }
```