

```
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     // Requiriements: 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 the 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 }
```