Project 1. You are to write (and test) software that rolls 6-sided dice.

Details. The software should be written as a potentially reusable component for other programs, such as games or simulations. Because not all games require two dice to be rolled (for example, some games, such as Trivial Pursuit, use a single die; while others, such as Risk, require three), the component should be able to "simultaneously" roll up to five dice at a time.

The user should be able to specify how many dice need to be rolled. The software should then return that many rolls of dice, independently generated.

The dice-rolling program need not be graphical in nature; it merely supplies randomly-generated numbers between 1 and 6, inclusive. The following two tables show some sample output (the printing part of this should be done separately from the dice-rolling routine itself; the dice-rolling routine should only return these values as output parameters to the calling routine):

***Single die (5 runs)*:**	***Two dice (5 runs)*:**
3	5 6
1	4 5
6	3 1
4	5 5
3	2 1

Testing. You should test your software, making sure it behaves as expected, to include (here's the tricky part) *making sure it sufficiently emulates randomness*. Think about what this means, and be sure to construct test cases that test this property of the output.

Documentation. Your final project submission should include your completed test plan (to include indications of whether each test case in the plan passed or failed). Also include output from test case runs as well as your source code as an appendix. If you have made any assumptions, document them and include those as well.

Remember, the test plan consists of several test cases, and their order of execution. Each test case has these components, as discussed in class: Inputs, Expected Outputs, Actual Output, and Test Case Result. Be sure to indicate whether or not the software has passed each test – this may seem like a redundant step, but how can a reviewer know for sure that a test has indeed been run, unless such annotations are made?

Note: If your testing discovers an error in your code, do not merely fix the error, and then rerun the test case. Document the error first, then fix the code. Your final turn-in should contain, in the Actual Results section, BOTH the results of the failed test, along with the results of the subsequent test. A brief note explaining what was done to correct the problem is considered a good practice, and should be employed throughout this course.

Remember, the bulk of your grade will depend upon the documentation – most especially the test plan – and not the code itself.

Project Milestones (due dates will be listed on Pilot):

HW 1 – Test Strategy. In paragraph form, briefly explain how you intend address the "tricky part" of this assignment (i.e., how you plan to prove your dice act randomly). No actual test cases are needed, but they can be included, if that makes things easier for you to explain.

HW 2 – Test Plan. This will be all the test cases in your test plan. You only need to document the test cases; you will not need to run the test cases or document the actual results for HW 2.

Project 1. The complete test plan, with annotated results, and other associated documentation as described above.