**Unit Testing**

The haunted building game was developed using Visual C# with a graphical user interface. No off the shelf game engine was used. We wanted to test most of the main functionalities that were provided in the final report. Specifically, we tested whether certain features where available or if they behaved as specified. The code was tested by running test suites on several functions in an attempt to determine how stable the program was. We didn’t want the game to crash because the user decided to enter invalid input.

The test suites detected several inputs that caused the program to crash or raise exceptions. We determined by single stepping in the debugger what was causing the exception and then fixed the problem. Our code works very well when used correctly but we discovered several situations that could cause errors, so documenting the exact result was important in this report. We used the method of white and black box testing for our test suites.

When running acceptance tests, we played the game from the perspective of the user and with a checklist of the functional and interface requirements, we reported what was satisfied and what was not. Most requirements were not achieved. However, the main requirements that made the game playable were accomplished. We were also able to provide a nice user interface for the game. Acceptance tests were done with the black box method.

**Inspection Testing**

Overall, each section of code that we inspected held up very well. Every function had correct and consistent naming conventions and was formatted appropriately. IF-ELSE statements and WHILE loops were always nested properly and correctly exited when a certain criteria was met. All of the functions work properly and repeatedly over multiple inputs and different game settings.

Using the *Generic Code Review Checklist* as our guide through code inspection, we were able to thoroughly and diligently examine a magnitude of possible issues with the code. The checklist contained some statements and questions that did not pertain to all the pieces of code we inspected. In the case of an inapplicable statement or question, we simply gave it a check as it technically ‘passed’ those specific criteria.

One reoccurring problem that was prevalent in all the code segments we inspected was that it was inadequately documented or commented. In some cases, there were little lines of comments listed before each function or method that briefly explained what the function did. This was only helpful to the creator of the method. The comments were often too vague and it would provide little insight to another party viewing the code for the first time. Documentation was also placed in areas that broke up the code too much. For example, in the monster code review the bulk of the documentation is in between IF-ELSE statements. This creates a large, unnecessary break between the condition statements and would be better suited if it were placed before the function itself.