**Test Report**

**File input** – per the instruction we are to assume that the files are formatted correctly, but may have trailing blank space at the end of the file.

Test #1: wrong file name - **PASS**

This application tests the Human component of the game and ensures that boards can be read in from a file.

Please choose a file that holds map data for the game: nofile.txt

An error occurred, please input another GameBoard file:

Test #2: another file name that exists, but is not a map: human.o – **FAIL (program should expect correctly formatted files though, so technically a pass per instructions)**

$Ddm

B

,Ll

!l"u

0P#$%&'4(T)t\*}

l+,!^

Where would you like to attack? (row column) (ex: A 1)

Input not recognized

Test #3: another map file with blank spaces at end – **FAIL (should change map input loop from reading until end of file to reading to the end of line 10, and to the end of the appropriate number of characters on line 10 (20 characters per line)**

This application tests the Human component of the game and ensures that boards can be read in from a file.

Please choose a file that holds map data for the game: blankmap.txt

0 1 2 3 4 5 6 7 8 9

A o o o o o o o o o o

B o o o o o o o o o o

C o o o o o o o o o o

D o o o o o o o o o o

E o o o o o o o o o o

F o o o o o o o o o o

G o o o o o o o o o o

H o o o o o o o o o o

I o o o o o o o o o o

J o o o o o o o o o o

@

**Attack Positions/Ranges –** the second prompt for user input occurs when the game asks the user for a char value and a number value for which position to attack

Test #1: Submitting a valid number and letter combination that results in a miss, both the first time and second time (that is if you’ve already selected that position before)- **PASS**

Where would you like to attack? (row column) (ex: A 1) A 1

0 1 2 3 4 5 6 7 8 9

A o M o o o o o o o o

B o o o o o o o o o o

C o o o o o o o o o o

D o o o o o o o o o o

E o o o o o o o o o o

F o o o o o o o o o o

G o o o o o o o o o o

H o o o o o o o o o o

I o o o o o o o o o o

J o o o o o o o o o o

Where would you like to attack? (row column) (ex: A 1)

Test #2: Submitting a valid number and letter combination that results in a hit – **PASS**

Where would you like to attack? (row column) (ex: A 1) E 0

You have hit a ship!

0 1 2 3 4 5 6 7 8 9

A o M o o o o o o o o

B o o o o o o o o o o

C o o o o o o o o o o

D o o o o o o o o o o

E H o o o o o o o o o

F o o o o o o o o o o

G o o o o o o o o o o

H o o o o o o o o o o

I o o o o o o o o o o

J o o o o o o o o o o

Test #3: Submitting a valid number and letter combination that results in a win **– PASS**

0 1 2 3 4 5 6 7 8 9

A o M o o o o o o o o

B o o o o o o o o o o

C o o o o o o o o o o

D H H H H H o o o o o

E H H H H H M o o o o

F M H H H H M o o o o

G o M M H H M o o o o

H o o M M o o o o o o

I o o o o o o o o o o

J o o o o o o o o o o

Where would you like to attack? (row column) (ex: A 1) H 4

You have hit a ship!

Congratulations, you have won the game

Test #4: Submitting a value that would be incorrect (outside the range of the board), either/or – or both, too large char value or too large number value - **PASS**

0 1 2 3 4 5 6 7 8 9

A o o o o o o o o o o

B o o o o o o o o o o

C o o o o o o o o o o

D o o o o o o o o o o

E o o o o o o o o o o

F o o o o o o o o o o

G o o o o o o o o o o

H o o o o o o o o o o

I o o o o o o o o o o

J o o o o o o o o o o

Where would you like to attack? (row column) (ex: A 1) X 10

Input not recognized

Where would you like to attack? (row column) (ex: A 1)

Test #5: Random string of text – **PASS**

Input not recognized

Where would you like to attack? (row column) (ex: A 1) FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Input not recognized

Where would you like to attack? (row column) (ex: A 1)

Test #6: Specific sets of weird characters – **FAIL (logical error in attackCoordinates() function in Proj1Aux.cpp – does not check for characters like ! # - and a few others, some other symbols like ^ ~ are caught. Char should be tested prior to sending to the player object’s map object to make sure its between an appropriate letter ASCII value 065-090 (or really 065-074 the size of the board) and the number value is 0-9 – or this check should be occurring correctly inside that map object so it doesn’t access vector or array values outside of the range.**

**(Ironically this bug made it easy and convenient to terminate the program for testing purposes while writing the program)**

Where would you like to attack? (row column) (ex: A 1) ###

terminate called after throwing an instance of 'std::out\_of\_range'

what(): vector::\_M\_range\_check

Aborted