TeamName:
Daniel Hwang
David Rust
Kyle Stewart

Chapter 3: Testing Framework, Test Cases, and How-To

Testing Framework:

frame.py:

This is the driver of the framework. It takes input from the *exampleTests.txt* input file and sends it through *work.c* to interact with the NetHack source code. After the all of the values from the functions being tested are returned, *frame.py* also compares them with the expected outputs also given in *exampleTests.txt* and writes the results (including: PASS/FAIL data, time elapsed, expected output, and actual output) to the *testlog.txt* file.

work.c:

This is the C code that connects frame.py to NetHack source code and also contains some source code from NetHack itself that is necessary for ***makeplural()*** and ***makesingular()*** in ***objnam.c,*** the source code being tested, to be call-able.

junk.sh:

This script is needs to be run after building NetHack to properly link ***objnam.c*** with its dependencies (and its dependencies' dependencies, and *their* dependencies' depen- … *this goes on for some time*).

exampleTests.txt:

Our input file for what functions to test, what arguments to send, and what the expected output is.

testlog.txt:

Our output file for the given tests' results (including: PASS/FAIL data, time elapsed, expected output, and actual output).

Test-Cases:

Here are five test cases in the format in which they would be given as input.
(NOTE: only commands prefixed with a "$" identifier will be run in frame.py)

```
$ makeplural human humans
Base case: PASS assures basic consonant-s format sing-to-plur conversion.

$ makeplural valkyrie valkyries
Base case: PASS assures basic vowel('e')-s format sing-to-plur conversion.

$ makeplural man men
```

TeamName:
Daniel Hwang
David Rust
Kyle Stewart

```
Badman: PASS assures basic -man to -men sing-to-plur conversion.

$ makeplural homunculus homunculi
-us to -i

$ makeplural larva larvae
-a to -ae
```