maze testplan

WO1 Clayton E. Williams

September 2023

1 Purpose

maze is a program that examines a map file, and prints the map with a shortest path between start and end, if it exists.
The purpose of this test plan is to provide a process of testing the functions of auxiliary libraries to gain reasonable assurance that the program exhibits desired behavior and does not crash on unexpected or invalid input.

2 Components

maze contains both automated and manual testing

2.1 Automated Tests - Test Suite

Automation of unit testing is provided through the make tool and can be run from the command line as:

```
$ make check
```

There are three test suites to test maze

- test_io_helper
- test_matrix
- test_llist

2.2 Test Cases

- test_validate_file_valid
  This test tests the validate_file() function with valid files and asserts the return value is 1.
- test_validate_file_invalid
  This test tests the validate_file() function with invalid files, either those that do not exist, directories, or chose such as /dev/null, and asserts they return 0.

- test_graph_create
  This tests the graph_create() function, asserting the returned pointer is not NULL

- test_get_set_graph_size
  This tests the get_set_graph_size() function, asserting the rows and columns of the graph are set appropriately based on the manual calculation of the ./data/valid_map.txt file.

- test_get_set_graph_size_invalid
  This tests get_set_graph_size() with a map that contains invalid characters and asserts the return value is 0.

- test_matrix_graph_create
  This tests the matrix_graph_create() function with a valid map, asserting the function returns 1, and that the start and null pointers are not NULL, and that the start and end pointer are equal to manually calculated matrix vertices.

- test_matrix_graph_create_invalid
  This tests the matrix_graph_create() function with maps that contain either none, or more than one start/end point, asserting the return value is 0.

- test_matrix_enrich
  Using previously validated maps, this function tests the matrix_enrich(), asserting the return value is 1 and that pointers and num_children are set appropriately.

- test_bfs
  This tests the bfs() function with a valid map, asserting the return value is 1.

- test_bfs_invalid
  This tests the bfs() function with a maze in which there is no path between the start and end point, ensuring the return value is 0.

Because of the modular design of the program, and the attempt to reduce iterations over the file or matrix, each function handles a small piece of the overall maze validation, and therefore, maze validation is cascading, with functions such as bfs() not being run unless matrix_validate_maze returns 1, which relies on matrix_enrich returning 1, and so on. Results of successful test run with make check:

```
100%: Checks: 10, Failures: 0, Errors: 0
test/test_io_helper.c:9:P:core:*curr++:0: Passed
test/test_io_helper.c:30:P:core:*curr++:0: Passed
test/test_matrix.c:36:P:core:*curr++:0: Passed
test/test_matrix.c:45:P:core:*curr++:0: Passed
```

```
test/test_matrix.c:52:P:core:*curr++:0: Passed
test/test_matrix.c:67:P:core:*curr++:0: Passed
test/test_matrix.c:75:P:core:*curr++:0: Passed
test/test_matrix.c:87:P:core:*curr++:0: Passed
test/test_matrix.c:97:P:core:*curr++:0: Passed
test/test_matrix.c:107:P:core:*curr++:0: Passed
```

2.3 Manual Tests - Valgrind

Manual testing of the program is to ensure there are no memory leaks or errors reported by valgrind:

```
$ make clean && make debug
$ valgrind ./maze <FILE> [OPTION]
==6793== HEAP SUMMARY:
==6793==     in use at exit: 0 bytes in 0 blocks
==6793==   total heap usage: 215 allocs, 215 frees, 19,864 bytes allocated
==6793==
==6793== All heap blocks were freed -- no leaks are possible
==6793==
==6793== For lists of detected and suppressed errors, rerun with: -s
==6793== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

2.4 Manual Tests - Stress Test

Manual stress testing of the program is provided to test against invalid file arguments, as well as valid files that contain valid and invalid mazes that fail throughout various functions and stages of the program execution. These test files can be found in /test/test_data, with /test/stress_test.txt being the controller. To run this stress test:

```
$make clean
$make
$while read -r line; do $line; done < test/test_data/tests.txt
```