Combination of all the testing suites and its outcome.

Blackbox testing : building test functions as we only know external sources of the program.

Code snippet can be found in **blackbox\_tester.c**  
#include <stdio.h>

#include "mapping.h"

// Test Case 1: populateMap()

void test\_populateMap()

{

struct Map map = populateMap();

// Verify the number of rows and columns

if (map.numRows == 25 && map.numCols == 25)

printf("populateMap test passed!\n");

else

printf("populateMap test failed!\n");

printf("\n");

}

// Test 2: printMap() // DEF1 and DEF2 has been resolved with new testing functions

// Helper function to check if the map coordinates are correct

int checkMapCoordinates(const struct Map\* map) {

int rowMax = map->numRows;

int colMax = map->numCols;

for (int r = 0; r < rowMax; r++) {

if (r + 1 < 1 || r + 1 > 25) {

printf("Row coordinate out of range: %d\n", r + 1);

return 0;

}

for (int c = 0; c < colMax; c++) {

if (c < 0 || c >= 25) {

printf("Column coordinate out of range: %d\n", c);

return 0;

}

}

}

return 1;

}

// Helper function to print the map

void testPrintMap(const struct Map\* map) {

char sym[] = " XB?G?.?Y?-?\*?+?P";

int rowMax = map->numRows;

printf("%4s", " ");

for (int c = 0; c < map->numCols; c++) {

printf("%c", 'A' + c);

}

printf("\n");

printf("%4s", " ");

for (int c = 0; c < map->numCols; c++) {

printf("-");

}

printf("\n");

for (int r = 1; r <= rowMax; r++) {

printf("%3d|", r);

for (int c = 0; c < map->numCols; c++) {

printf("%c", sym[map->squares[r - 1][c]]);

}

printf("\n");

}

}

// Test printMap function

void test\_printMap() {

// Test case 1: base1 = 1, alphaCols = 1

struct Map map1 = populateMap();

printf("\*\*\* PrintMap Test case 1:\n");

printMap(&map1, 1, 1);

printf("\n");

if (checkMapCoordinates(&map1))

printf(": printMap Test case 1 passed!\n\n");

else

printf(": printMap Test case 1 failed!\n\n");

// Test case 2: base1 = 10, alphaCols = 0

struct Map map2 = populateMap();

printf("\*\*\* PrintMap Test case 2:\n");

printMap(&map2, 10, 0);

printf("\n");

if (!checkMapCoordinates(&map2))

printf(": printMap Test case 2 passed!\n\n");

else

printf(": printMap Test case 2 failed!\n\n");

// Test case 3: base1 = 20, alphaCols = 1

struct Map map3 = populateMap();

printf("\*\*\* PrintMap Test case 3:\n");

printMap(&map3, 20, 1);

printf("\n");

if (!checkMapCoordinates(&map3))

printf(": printMap Test case 3 passed!\n\n");

else

printf(": printMap Test case 3 failed!\n\n");

}

int main() {

test\_populateMap();

test\_printMap();

return 0;

}

We have done necessary debugging on parts that does not have passed tests.

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

**White Box Testing:** White box testing was performed as if we know all the internal area of this program. Code snippets can be found in **whitebox\_finder\_tester.c** and **whitebox\_mapping\_tester.c.**  We have tested whitebox strategy on the function implementation that we did and the mapping function that was given for this project to start.

**Whitebox\_fidner\_tester.c :**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "finder.h"

#include "mapping.h"

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

void test\_hasDestination();

void test\_isTruckOverloaded();

void test\_isBoxSizeExceeded();

void test\_validCargo();

int main()

{

test\_hasDestination();

test\_isTruckOverloaded();

test\_isBoxSizeExceeded();

test\_validCargo();

return 0;

}

// Test cases for hasDestination

// August 8 fixed from

/\*

void test\_hasDestination()

{

// Define and initialize the necessary structures and variables for testing

struct Shipment shipment;

shipment.destination.row = 5;

shipment.destination.col = 5;

struct Route route;

// Initialize the route as required for testing

// Test the function

int result = hasDestination(&route, shipment);

// Print the results

printf("test\_hasDestination ");

if (result == 1)

{

printf("test passed!\n");

}

else

{

printf("test failed!\n");

}

}

\*/

void test\_hasDestination()

{

struct Shipment shipment;

shipment.destination.row = 5;

shipment.destination.col = 5;

struct Route route;

route.numPoints = 3;

route.points[0] = (struct Point){ 3, 3 };

route.points[1] = (struct Point){ 4, 4 };

route.points[2] = (struct Point){ 5, 5 };

// Test the function

int result = hasDestination(&route, shipment);

// Print the results

printf("test\_hasDestination ");

if (result == 1)

{

printf("test passed!\n");

}

else

{

printf("test failed!\n");

}

}

// Test cases for isTruckOverloaded

void test\_isTruckOverloaded()

{

// Define and initialize the necessary structures and variables for testing

struct Truck truck;

// Initialize the truck as required for testing

struct Shipment ship;

// Initialize the shipment as required for testing

// Test the function

int result = isTruckOverloaded(truck, ship);

// Print the results

printf("test\_isTruckOverloaded ");

if (result == 1)

{

printf("test passed!\n");

}

else

{

printf("test failed!\n");

}

}

// Test cases for isBoxSizeExceeded

void test\_isBoxSizeExceeded()

{

// Define and initialize the necessary structures and variables for testing

struct Truck truck;

// Initialize the truck as required for testing

float boxSize = 5.0; // Set the boxSize value as required for testing

// Test the function

int result = isBoxSizeExceeded(truck, boxSize);

// Print the results

printf("test\_isBoxSizeExceeded ");

if (result == 1)

{

printf("test passed!\n");

}

else

{

printf("test failed!\n");

}

}

// Test cases for vaildCargo

void test\_validCargo()

{

// Define and initialize the necessary structures and variables for testing

float boxSize = 0.5; // Set the boxSize value as required for testing

// Test the function

int result = validCargo(boxSize);

// Print the results

printf("test\_validCargo ");

if (result == 1)

{

printf("test passed!\n");

}

else

{

printf("test failed!\n");

}

}

**Whitebox\_mapping\_tester.c**

#include <stdio.h>

#include "mapping.h"

// Test case for add\_route()

// August 8 fixed from

/\*

void test\_addRoute()

{

struct Map map = populateMap();

struct Map mapWithRoute;

struct Route route = { 'X', 1, 1, 5, 'E' };

mapWithRoute = addRoute(&map, &route);

// Verify if the first character of the route is correctly added to the map

if (mapWithRoute.squares[1][1] == 'X')

printf("addRoute test case1 passed!\n");

else

printf("addRoute test case1 failed!\n");

}

\*/

void test\_addRoute()

{

struct Map map = populateMap();

struct Map mapWithRoute;

struct Route route = { 0 }; // Initialize route with default values

route.numPoints = 4;

route.routeSymbol = 'X';

route.points[0] = (struct Point){ 1, 1 };

route.points[1] = (struct Point){ 1, 2 };

route.points[2] = (struct Point){ 1, 3 };

route.points[3] = (struct Point){ 1, 4 };

mapWithRoute = addRoute(&map, &route);

// Verify if the route symbol is correctly added to the map

if (mapWithRoute.squares[1][1] == 'X')

printf("addRoute test case passed!\n");

else

printf("addRoute test case failed!\n");

}

int main()

{

test\_addRoute();

return 0;

}