**Test Report Summary**

**Summary of the Testing Process:**

We conducted black and white testing for each function that could cause problems, and mainly investigated each branch point. I used debugging (when using values ​​from the console) and assertions to make sure the function worked the way it should. After confirming that each function was working, integration testing was also conducted to call other functions within the function. A total of 183 tests were conducted, and all were successful except for three cases that failed intentionally.

**RESULT of Testing:**

Assertions: 60PASS, 3FAIL (planned fail)

Docs : 120PASS, 0FAIL

Descriptions of failed cases are added below.

**Time Taken for Testing:**

Total 3 members

Testing conducted for 4 out of 6 weeks (entire project)

Spent time: planning (3hours), Develop cases (about 6hours each), implementing (about 4 hours each- black, white, integration),

**Test Coverage:**

The test suite covers the code for both main.c, mapping.c and shipping.c. It covers all the lines of the code. Unit testing module made from given and custom function that we made.

**Total Coast:**

No coast !!! it’s group work all made by ourselves.

**Unit test.cpp & bug report:**

🡪 Testing implemented with assertions, omitted for integration tests that use data values ​​from the console.

**These three FAIL cases are normal behavior (which means they’re virtually pass). I explain the reason below. (we don’t have to fix the code because of them).**

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test type | Description | Result |
| ModuleInitialize |  | Verifying test module initialization | PASS |
| ModuleCleanup |  | Verifying test module cleanup | PASS |
| ClassInitialize |  | Verifying test class initialization | PASS |
| ClassCleanup |  | Verifying test class cleanup | PASS |
| WTS\_TestMethod1 | Blackbox | Validating function ‘checkWeight’ When input is within the maximum value. | PASS |
| WTS\_TestMethod2 | Blackbox | Validating function ‘checkWeight’ When input is 0. | PASS |
| WTS\_TestMethod3 | Whitebox | Validating function ‘checkWeight’ When input is negative value. | PASS |
| WTS\_TestMethod4 | Whitebox | Validating function ‘checkWeight ‘When input exceeds the maximum value. | PASS |
| WTS\_TestMethod5 | Whitebox | Validating function ‘checkWeight’ When input is double type. | PASS |
| WTS\_TestMethod6 | Whitebox | Validating function ‘checkWeight’ When input is invalid/NULL. | PASS |
| TPM\_Testcase1 | Whitebox(integration) | Validating function ‘getPossibleMoves’ If all square, checkRow, and check Weight functions are passed with input. | PASS |
| TPM\_Testcase2 | Whitebox(integration) | Validating function ‘getPossibleMoves’ if checkRow, and check Weight functions are passed but square not passed with input. | PASS |
| TPM\_Testcase3 | Whitebox(integration) | Validating function ‘getPossibleMoves’ when Map input is Null | FAIL |
| TPM\_Testcase4 | Whitebox(integration) | Validating function ‘getPossibleMoves’ if map point including 0 value | PASS |
| STF\_TestMethod1 | Whitebox(integration) | Validating function ‘shortestPath’ if start, end point is in valiid spot, also map row, col 5each | PASS |
| STF\_TestMethod2 | Whitebox(integration) | Validating function ‘shortestPath’ if start, end point is in valiid spot, also map row, col 5each. But each spot ordered opposite. | PASS |
| STF\_TestMethod3 | Whitebox(integration) | Validating function ‘shortestPath’ if start, end point is in valiid spot, but both are same, also map row, col 5each. | PASS |
| STF\_TestMethod4 | Whitebox(integration) | Validating function ‘shortestPath’ if start, end point is in valiid spot, also map row, col 5each. But instance of struct Map is nullptr. | FAIL |
| CST\_TestMethod1 | Blackbox | Validating function ‘checkSize’ When input is matched with small box size. | PASS |
| CST\_TestMethod2 | Blackbox | Validating function ‘checkSize’ When input is matched with mideum box size. | PASS |
| CST\_TestMethod3 | Blackbox | Validating function ‘checkSize’ When input is matched with large box size. | PASS |
| CST\_TestMethod4 | Blackbox | Validating function ‘checkSize’ When input is 0. | PASS |
| CST\_TestMethod5 | Blackbox | Validating function ‘checkSize’ When input is invalid/ NULL. | PASS |
| CMS\_TestMethod1 | Blackbox | Validating function ‘checkPoint’ When input row, cols are within 25 and positive. | PASS |
| CMS\_TestMethod2 | Blackbox | Validating function ‘checkPoint’ When input row is negative. | PASS |
| CMS\_TestMethod3 | Blackbox | Validating function ‘checkPoint’ When input col is negative. | PASS |
| CMS\_TestMethod4 | Whitebox | Validating function ‘checkPoint’ When one of the input is 0. | PASS |
| CMS\_TestMethod5 | Whitebox | Validating function ‘checkPoint’ When one of the input is NULL. | PASS |
| APT\_TestMethod1 | Blackbox(integration) | Validating function ‘checkAll’ when size value is smaller than minimum. | PASS |
| APT\_TestMethod2 | Blackbox(integration) | Validating function ‘checkAll’ when weight value is smaller than minimum. | PASS |
| APT\_TestMethod3 | Blackbox(integration) | Validating function ‘checkAll’ when point value is not on the range. | PASS |
| APT\_TestMethod4 | Whitebox(integration) | Validating function ‘checkAll’ when All valid values but couldnt pass blocked space. | PASS |
| APT\_TestMethod5 | Whitebox(integration) | Validating function ‘checkAll’ when size exceed maximum. | PASS |
| APT\_TestMethod6 | Whitebox(integration) | Validating function ‘checkAll’ when weight exceed maximum. | PASS |
| APT\_TestMethod8 | Whitebox(integration) | Validating function ‘checkAll’ when Point is NULL (no value). | PASS |
| TEP\_TestMethod1 | Whitebox | Validating function ‘eqPt’ when valid spot and matched values. | PASS |
| TEP\_TestMethod2 | Whitebox | Validating function ‘eqPt’ when invalid point but matched. | PASS |
| TEP\_TestMethod3 | Whitebox | Validating function ‘eqPt’ when both doesnt match. | PASS |
| TEP\_TestMethod4 | Whitebox | Validating function ‘eqPt’ when only one matches. | PASS |
| TEP\_TestMethod5 | Whitebox | Validating function ‘eqPt’ when uncompleted input. | PASS |
| TNR\_TestMethod1 | Blackbox(integration) | Validating function ‘getNumRows’ with nullptr. | PASS |
| TNR\_TestMethod2 | Blackbox(integration) | Validating function ‘getNumRows’ with all valid input but not passing square. | PASS |
| TNR\_TestMethod3 | Blackbox(integration) | Validating function ‘getNumRows’ with all valid input and passing square. | PASS |
| TNR\_TestMethod6 | Whitebox(integration) | Validating function ‘getNumRows’ with all valid input but empty square array(not Null). | PASS |
| TCP\_TestMethod1 | Blackbox | Validating function ‘calculatePercent’ with matched input types. | PASS |
| TCP\_TestMethod2 | Blackbox | Validating function ‘calculatePercent’ with double input types. | PASS |
| TCP\_TestMethod4 | Blackbox | Validating function ‘calculatePercent’ with negative input type. | PASS |
| TCP\_TestMethod6 | Blackbox | Validating function ‘calculatePercent’ with divided by 0 (infinity). | PASS |
| TCP\_TestMethod7 | Whitebox | Validating function ‘calculatePercent’ with int,int,int. | PASS |
| TCP\_TestMethod8 | Whitebox | Validating function ‘calculatePercent’ with double type but cast the result. | PASS |
| TCP\_TestMethod9 | Whitebox | Validating function ‘calculatePercent’ with 0. | PASS |
| TCP\_TestMethod10 | Whitebox | Validating function ‘calculatePercent’ with empty value(NULL). | PASS |
| TMT\_TestMethod1 | Whitebox(integration) | Validating function ‘findTruck’ with all input and outputs are matched type, and matched in truck A. | PASS |
| TMT\_TestMethod2 | Whitebox(integration) | Validating function ‘findTruck’ with All matched to A but invalid adress spot. | PASS |
| TMT\_TestMethod4 | Whitebox(integration) | Validating function ‘findTruck’ when the value on "val" is double not integer as pre declared but narrowed so case B matched. | PASS |
| TMT\_TestMethod5 | Whitebox(integration) | Validating function ‘findTruck’ with intentionally causing overflow. | PASS |
| TMT\_TestMethod6 | Whitebox(integration) | Validating function ‘findTruck’ with height exceed the value but matched to truck B. | PASS |
| TMT\_TestMethod7 | Whitebox(integration) | Validating function ‘findTruck’ with NULL spot. | PASS |
| APR\_Testcase1 | Blackbox(integration) | Validating function ‘AddpointToRoute’ with 0 route member data’s. | PASS |
| APR\_Testcase2 | Blackbox(integration) | Validating function ‘AddpointToRoute’ with 0 route member data’s. but not casting result. | PASS |
| APR\_Testcase3 | Blackbox(integration) | Validating function ‘AddpointToRoute’ when numPoint is not 0. | PASS |
| APR\_Testcase4 | Whitebox(integration) | Validating function ‘AddpointToRoute’ when argument structure is nullptr. | FAIL |
| APR\_Testcase5 | Whitebox(integration) | Validating function ‘AddpointToRoute’ when row,col value is Char\_MAX. | PASS |

**Bug report**: Functions that fail have something in common: there is an instance of Map in the function's argument and this is set to NULL. Because of the basic form of the structure Map, the above functions that use console values ​​cannot be used. In other words, these test cases were to check whether the "error" occurred normally, and if the assertion is set to ExpectException, the function can be passed. So, you can think these cases was intentionally made to be fail.

* In the case of testing the functions findTruck, checkAll, onTheRoute, and FindBestRoute commented out in the code, it was impossible to implement them with assertions, so a rough version was implemented and commented out.

**Testing Docs:**

Rest of cases in Docs, Acceptance test report exist separately.

< Simple Organized by identifier>

T01xx: Testing function checkWeight T02xx: Testing function checkSize

T03xx: Testing function checkpoint T04xx: Testing function checkAll

T05xx: Testing function eqPt T06xx: Testing function getNumRows

T07xx: Testing function onTheRoute T08xx: Testing function findBestRoute

T09xx: Testing function chooseTruck T10xx: Testing function calculatePercent

T11xx: Testing function findTruck

**All cases are passed.**

|  |  |  |  |
| --- | --- | --- | --- |
| TEST TYPE | TEST ID | DESCRIPTION | **PASS / FAIL** |
| Blackbox | T0101 | Weight is positive and within maximum | PASS |
| Blackbox | T0102 | Weight is 0 | PASS |
| Blackbox | T0103 | Negative weight value | PASS |
| Blackbox | T0104 | Exceeding maximum weight | PASS |
| Blackbox | T0105 | Overflow with exceeding 2147483647(max num with int type) | PASS |
| Blackbox | T0106 | Case weight having float point value when removes become 0 | PASS |
| Blackbox | T0107 | Argument value is NULL | PASS |
| Blackbox | T0108 | Special char used as argument | PASS |
| Whitebox | T0109 | Negative weight case | PASS |
| Whitebox | T0110 | Edge value | PASS |
| Whitebox | T0111 | More than max edge | PASS |
| Whitebox | T0112 | Case when validation aborted | PASS |
| Blackbox | T0201 | Argument same size as the small | PASS |
| Blackbox | T0202 | Argument same size as the mid | PASS |
| Blackbox | T0203 | Argument same size as large | PASS |
| Blackbox | T0204 | Case value pass negative size | PASS |
| Blackbox | T0205 | Case input is NULL | PASS |
| Blackbox | T0206 | Input is unrelated type (char) | PASS |
| Blackbox | T0207 | Positive but not matched | PASS |
| Whitebox | T0208 | Value between small and mid | PASS |
| Whitebox | T0209 | Value same with small when narrowed | PASS |
| Whitebox | T0210 | 0 value inserted | PASS |
| Blackbox | T0301 | row and col are both within the valid value of 25 | PASS |
| Blackbox | T0302 | row is out of range (negative number) at location | PASS |
| Blackbox | T0303 | col is out of range (negative number) at location | PASS |
| Blackbox | T0304 | start of coordinates will be recognized from 0 with one value 0 | PASS |
| Blackbox | T0305 | Invalid data (char type) | PASS |
| Blackbox | T0306 | One of the member data missing (struct point’s) | PASS |
| Whitebox(integration) | T0307 | Both row, col within range but 1 value when call square. | PASS |
| Whitebox | T0308 | Row isn’t in range | PASS |
| Whitebox | T0309 | Col isn’t in range | PASS |
| Whitebox | T0310 | Case row is negative | PASS |
| Whitebox(integration) | T0311 | Bot row, col within the range and 0 value when call square | PASS |
| Whitebox(integration) | T0312 | Cols not in the range with 1 square value | PASS |
| Blackbox | T0401 | Test with a invalid size (less than minimum value) | PASS |
| Blackbox | T0402 | Test with a invalid weight (less than minimum value) | PASS |
| Blackbox | T0403 | Test with a invalid point (not within the scope) | PASS |
| Blackbox | T0404 | All parameters are valid. | PASS |
| Blackbox(Integration) | T0405 | Test with a invalid size and weight, checking the break | PASS |
| Blackbox(Integration) | T0406 | Test with a invalid weight and point | PASS |
| Blackbox(Integration) | T0407 | Invalid input (no int type as weight) | PASS |
| Blackbox(Integration) | T0408 | Invalid input (empty input struct point) | PASS |
| Whitebox | T0409 | All parameters are valid. | PASS |
| Whitebox | T0410 | Case weight is invalid | PASS |
| Whitebox | T0411 | Case Point is invalid | PASS |
| Whitebox(Integration) | T0412 | Change the value of column to be invalid | PASS |
| Whitebox(Integration) | T0413 | All values are valid but 0value returned by square function | PASS |
| Whitebox(Integration) | T0414 | All values are valid and 1value returned by square function | PASS |
| Whitebox(Integration) | T0415 | When size and point is invalid | PASS |
| Blackbox | T0501 | the member data values ​​of both objects are the same | PASS |
| Blackbox | T0502 | invalid value of false but both data is same | PASS |
| Blackbox | T0503 | row and col of the two objects do not match | PASS |
| Blackbox | T0504 | Negative value and doesn’t match | PASS |
| Blackbox | T0505 | One value matches | PASS |
| Blackbox | T0506 | All value matching with char input | PASS |
| Blackbox | T0507 | The member data of p2 is defined as int, and if one is left blank check for exception case. | PASS |
| Whitebox | T0508 | If the member data values ​​of both objects are the same | PASS |
| Whitebox | T0509 | Row of struct point is invalid but matches | PASS |
| Whitebox | T0510 | Col of struct point is invalid but matches | PASS |
| Whitebox | T0511 | Case when nullptr is included as input | PASS |
| Blackbox | T0601 | Test with a empty map | PASS |
| Blackbox(integration) | T0602 | Test with a map with 1 row checking square | PASS |
| Blackbox(integration) | T0603 | Test with a map with 2 rows checking square | PASS |
| Blackbox(integration) | T0604 | Test with a map with 3 rows checking square | PASS |
| Whitebox | T0605 | Couldn’t go in validation with nullptr on map | PASS |
| Whitebox | T0606 | No square Data and map stuct is valid and row value is 2 | PASS |
| Whitebox(integration) | T0607 | No square Data and map stuct is valid and row value is MAX | PASS |
| Whitebox(integration) | T0608 | Squares uninitialized(empty arr) passing validation | PASS |
| Whitebox(integration) | T0609 | valid spot square(10,10) and not null points | PASS |
| Whitebox(integration) | T0610 | invalid spot square(10,10) but valid points member and square | PASS |
| Blackbox(Integration) | T0701 | are not in any points, no matching among them. | PASS |
| Blackbox(Integration) | T0702 | passing route with spot cols ->0, matched all and also points number. But exist matching spot | PASS |
| Blackbox(Integration) | T0703 | numPoint doesn’t matches, and matched index are bigger than it, spot is valid too. | PASS |
| Blackbox(Integration) | T0704 | Num points aren’t matches with point cases and matched index are bigger than it | PASS |
| Blackbox(Integration) | T0705 | Matched but matched index exceed numPoint | PASS |
| Blackbox(Integration) | T0706 | Matched point should be 0,0 no point exists but numPoint is bigger than declared point so case if it makes empty arr. | PASS |
| Blackbox(Integration) | T0707 | When passing invalid char or nullptr as point. | PASS |
| Blackbox | T0801 | no available route, the route array is empty. | PASS |
| Blackbox | T0802 | Delivery point is not in array | PASS |
| Whitebox(integration) | T0803 | Delivery point is in array but case when the values of col,row returns 1 when passing square | PASS |
| Whitebox(integration) | T0804 | Delivery point is in array and the values of col,row returns 0 when passing square best route is on first index. | PASS |
| Whitebox(integration) | T0805 | Delivery point is in array and the values of col,row returns 0 when passing square best route is on second index. | PASS |
| Blackbox | T0806 | Number of route is less than argument | PASS |
| Blackbox | T0807 | Same as T0806 but total point value is char type(invalid) | PASS |
| Blackbox | T0901 | Passing with less weight in index 1’s ship | PASS |
| Blackbox(integration) | T0902 | Passing with full weight in index 1’s ship | PASS |
| Blackbox(integration) | T0903 | Same level of occupation | PASS |
| Blackbox(integration) | T0904 | Both truck has same volume and weight without exceeding. | PASS |
| Blackbox | T0905 | Both weight has exceed the max. | PASS |
| Whitebox(integration) | T0906 | When add ship 1,2 weight no more than MAX | PASS |
| Whitebox(integration) | T0907 | Exceed max when added | PASS |
| Whitebox | T0908 | When doesn’t exceed but ship1’s symbol is missing | PASS |
| Whitebox(integration) | T0909 | Empty weight data of ship2 | PASS |
| Whitebox(integration) | T0910 | Invalid weight – data of ship 2 | PASS |
| Blackbox | T1001 | Dividing with int positive values | PASS |
| Blackbox | T1002 | When val(number divided) is 0 passing test | PASS |
| Blackbox | T1003 | When invalid val (char type) | PASS |
| Blackbox | T1004 | When val is double type narrowing without cast | PASS |
| Blackbox | T1005 | when max is missing | PASS |
| Blackbox | T1006 | When max(number dividing) is 0 causing infinity | PASS |
| Blackbox | T1007 | When val is negative | PASS |
| Whitebox | T1008 | Checking all valid value because it’s percent | PASS |
| Whitebox | T1009 | Case Val is incorrect input | PASS |
| Whitebox | T1010 | Check if it works on narrowing (cast) | PASS |
| Whitebox | T1011 | Testing infinity case | PASS |
| Whitebox | T1012 | Check with negative val (double type) | PASS |
| Blackbox(integration) | T1101 | When point of destination is valid (close to index0)  Also having the symbol | PASS |
| Blackbox | T1102 | When point of destination is not valid | PASS |
| Blackbox | T1103 | When invalid weight value passed(char) | PASS |
| Blackbox | T1104 | When destination is valid but ship’s symbol is missing | PASS |
| Blackbox | T1105 | When the value is bigger than MAX value that int type can handle | PASS |
| Blackbox(integration) | T1106 | When point of destination is valid (close to index1)  Also having the symbol, passing the square point too | PASS |
| Blackbox | T1107 | Passing findTruck function with Empty spot point value | PASS |
| Blackbox(integration) | T1108 | When point of destination is valid (close to index2)  Also having the symbol, passing the square point too | PASS |
| Whitebox | T1109 | Case Volume exceed Max | PASS |
| Whitebox(integration) | T1110 | all valid inputs, square, case initial value is green and weight isn’t exceeded | PASS |
| Whitebox(integration) | T1111 | all valid inputs, square, case initial value is green and weight is exceeded | PASS |
| Whitebox(integration) | T1112 | all valid inputs, square, case initial value is green and weight is exceeded, blue’s weight is exceeded | PASS |
| Whitebox(integration) | T1113 | Case if all inputs are valid but square returns 1 with that point | PASS |