|  |  |
| --- | --- |
|  |  |
| MAZE GAME STAGE 4 |  |
|  |  |
|  | Software Design And Development |
|  | 36238194 2022 |

Table of Contents

[Testing 3](#_Toc106867041)

[1 Subroutine: SortHighscores 3](#_Toc106867042)

[Code 3](#_Toc106867043)

[Test Data 4](#_Toc106867044)

[Driver Code 11](#_Toc106867045)

[Test Result 18](#_Toc106867046)

[Discussion 18](#_Toc106867047)

[2 Function: ValidateSeed 19](#_Toc106867048)

[Code 19](#_Toc106867049)

[Test Data 20](#_Toc106867050)

[Driver Code 21](#_Toc106867051)

[Test Result 21](#_Toc106867052)

[Discussion 21](#_Toc106867053)

[3 Function: ValidateName 22](#_Toc106867054)

[Code 22](#_Toc106867055)

[Test Data 23](#_Toc106867056)

[Driver Code 24](#_Toc106867057)

[Test Result 24](#_Toc106867058)

[Discussion 24](#_Toc106867059)

[Testing on different specifications 25](#_Toc106867060)

[Description 25](#_Toc106867061)

[Evidence and Results 26](#_Toc106867062)

[Testing with different install locations 27](#_Toc106867063)

[Description 27](#_Toc106867064)

[Evidence 1 27](#_Toc106867065)

[Results 1 27](#_Toc106867066)

[Evidence 2 28](#_Toc106867067)

[Results 2 28](#_Toc106867068)

[Changes required to program 28](#_Toc106867069)

[Evaluation 29](#_Toc106867070)

[Design Specifications 29](#_Toc106867071)

[User’s Perspective 29](#_Toc106867072)

[Screens 33](#_Toc106867073)

[User Evaluation 39](#_Toc106867074)

[Questions 39](#_Toc106867075)

[Evidence of survey 39](#_Toc106867076)

[Overall survey summary 39](#_Toc106867077)

[Individual Response Summary 40](#_Toc106867078)

# Testing

## 1 Subroutine: SortHighscores

### Code

|  |
| --- |
| *' NESA standard bubble sort on one index of arrHighscores, using the gameTime property, in ascending order*  Public Sub SortHighscores(ByRef arrHighscores As Highscore()(), ByVal sizeToSort As Integer)  Dim last As Integer = arrHighscores(sizeToSort).Length - 1  Dim swapped As Boolean = True  Dim i As Integer = 0  While swapped  swapped = False  i = 0  While i < last  If arrHighscores(sizeToSort)(i).gameTime > arrHighscores(sizeToSort)(i + 1).gameTime Then  Swap(arrHighscores, sizeToSort, i, i + 1)  swapped = True  End If  i = i + 1  End While  last = last - 1  End While End Sub *' Swap was included as it's a dependency of this function* Public Sub Swap(arrHighscores As Highscore()(), size As Integer, idxA As Integer, idxB As Integer)  Dim temp As Highscore = arrHighscores(size)(idxA)  arrHighscores(size)(idxA) = arrHighscores(size)(idxB)  arrHighscores(size)(idxB) = temp End Sub |

### 

### Test Data

| **Variable Name** | **Variable Value** | **Expected Output** | **Reason for Inclusion** |
| --- | --- | --- | --- |
| arrHighscores | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 50 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 41 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 22 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 52 | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 50 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 41 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 22 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 52 | Data Set 1: The array is already sorted with all distinct values for playerName and gameTime. |
| arrHighscores | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 20 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 40 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 21 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 21 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 12 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 12 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 52 | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 20 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 40 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 21 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 21 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 12 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 12 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 52 | Data Set 2: The array is already sorted with all distinct values for playerName, but some duplicate values for gameTime. |
| arrHighscores | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 50 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 10 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 40 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 20 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 30 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 31 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 41 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 11 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 51 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 21 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 52 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 22 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 32 | arrHighscores(0)(0).playerName = "Bob" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Damon" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Edvin" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Cheryl" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Alfred" arrHighscores(0)(4).gameTime = 50 arrHighscores(1)(0).playerName = "Henry" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "Jacob" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Frank" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "George" arrHighscores(1)(3).gameTime = 41 arrHighscores(1)(4).playerName = "Isaac" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Monty" arrHighscores(2)(1).gameTime = 22 arrHighscores(2)(2).playerName = "Olivia" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Ludwig" arrHighscores(2)(4).gameTime = 52 | Data Set 3: The array contains all distinct values for gameTime and playerName but has been randomly shuffled. |
| arrHighscores | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 50 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 40 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 20 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 10 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 51 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 41 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 21 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 11 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 52 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 42 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 22 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 12 | arrHighscores(0)(0).playerName = "Edvin" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Damon" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Bob" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Alfred" arrHighscores(0)(4).gameTime = 50 arrHighscores(1)(0).playerName = "Jacob" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "Isaac" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "George" arrHighscores(1)(3).gameTime = 41 arrHighscores(1)(4).playerName = "Frank" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Olivia" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Nathan" arrHighscores(2)(1).gameTime = 22 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Ludwig" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Kevin" arrHighscores(2)(4).gameTime = 52 | Data Set 4: The array is already sorted with all distinct values for gameTime and playerName but is in reverse order. |
| arrHighscores | arrHighscores(0)(0).playerName = "Bob" arrHighscores(0)(0).gameTime = 20 arrHighscores(0)(1).playerName = "Cheryl" arrHighscores(0)(1).gameTime = 30 arrHighscores(0)(2).playerName = "Damon" arrHighscores(0)(2).gameTime = 40 arrHighscores(0)(3).playerName = "Edvin" arrHighscores(0)(3).gameTime = 50 arrHighscores(0)(4).playerName = "Alfred" arrHighscores(0)(4).gameTime = 10 arrHighscores(1)(0).playerName = "George" arrHighscores(1)(0).gameTime = 21 arrHighscores(1)(1).playerName = "Henry" arrHighscores(1)(1).gameTime = 31 arrHighscores(1)(2).playerName = "Isaac" arrHighscores(1)(2).gameTime = 41 arrHighscores(1)(3).playerName = "Jacob" arrHighscores(1)(3).gameTime = 51 arrHighscores(1)(4).playerName = "Frank" arrHighscores(1)(4).gameTime = 11 arrHighscores(2)(0).playerName = "Ludwig" arrHighscores(2)(0).gameTime = 22 arrHighscores(2)(1).playerName = "Monty" arrHighscores(2)(1).gameTime = 32 arrHighscores(2)(2).playerName = "Nathan" arrHighscores(2)(2).gameTime = 42 arrHighscores(2)(3).playerName = "Olivia" arrHighscores(2)(3).gameTime = 52 arrHighscores(2)(4).playerName = "Kevin" arrHighscores(2)(4).gameTime = 12 | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 50 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 41 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 51 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 22 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 52 | Data Set 5: The array is already sorted with all distinct values for playerName, however the first-placed player has been moved to last. |
| arrHighscores | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 20 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 30 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 40 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 10 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 21 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 31 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 41 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 11 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 22 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 32 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 42 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 12 | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 10 arrHighscores(0)(1).playerName = "Edvin" arrHighscores(0)(1).gameTime = 10 arrHighscores(0)(2).playerName = "Bob" arrHighscores(0)(2).gameTime = 20 arrHighscores(0)(3).playerName = "Cheryl" arrHighscores(0)(3).gameTime = 30 arrHighscores(0)(4).playerName = "Damon" arrHighscores(0)(4).gameTime = 40 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 11 arrHighscores(1)(1).playerName = "Jacob" arrHighscores(1)(1).gameTime = 11 arrHighscores(1)(2).playerName = "George" arrHighscores(1)(2).gameTime = 21 arrHighscores(1)(3).playerName = "Henry" arrHighscores(1)(3).gameTime = 31 arrHighscores(1)(4).playerName = "Isaac" arrHighscores(1)(4).gameTime = 41 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 12 arrHighscores(2)(1).playerName = "Olivia" arrHighscores(2)(1).gameTime = 12 arrHighscores(2)(2).playerName = "Ludwig" arrHighscores(2)(2).gameTime = 22 arrHighscores(2)(3).playerName = "Monty" arrHighscores(2)(3).gameTime = 32 arrHighscores(2)(4).playerName = "Nathan" arrHighscores(2)(4).gameTime = 42 | Data Set 6: The array is already sorted with mostly distinct values for playerName, however a duplicate of the first-placed player has been placed in last. |
| arrHighscores | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 15 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 15 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 15 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 15 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 15 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 25 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 25 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 25 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 25 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 25 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 35 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 35 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 35 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 35 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 35 | arrHighscores(0)(0).playerName = "Alfred" arrHighscores(0)(0).gameTime = 15 arrHighscores(0)(1).playerName = "Bob" arrHighscores(0)(1).gameTime = 15 arrHighscores(0)(2).playerName = "Cheryl" arrHighscores(0)(2).gameTime = 15 arrHighscores(0)(3).playerName = "Damon" arrHighscores(0)(3).gameTime = 15 arrHighscores(0)(4).playerName = "Edvin" arrHighscores(0)(4).gameTime = 15 arrHighscores(1)(0).playerName = "Frank" arrHighscores(1)(0).gameTime = 25 arrHighscores(1)(1).playerName = "George" arrHighscores(1)(1).gameTime = 25 arrHighscores(1)(2).playerName = "Henry" arrHighscores(1)(2).gameTime = 25 arrHighscores(1)(3).playerName = "Isaac" arrHighscores(1)(3).gameTime = 25 arrHighscores(1)(4).playerName = "Jacob" arrHighscores(1)(4).gameTime = 25 arrHighscores(2)(0).playerName = "Kevin" arrHighscores(2)(0).gameTime = 35 arrHighscores(2)(1).playerName = "Ludwig" arrHighscores(2)(1).gameTime = 35 arrHighscores(2)(2).playerName = "Monty" arrHighscores(2)(2).gameTime = 35 arrHighscores(2)(3).playerName = "Nathan" arrHighscores(2)(3).gameTime = 35 arrHighscores(2)(4).playerName = "Olivia" arrHighscores(2)(4).gameTime = 35 | Data Set 7: The array contains all distinct values for playerName but each value of gameTime is the same. |

Note that when referring to 'the array' in 'Reason for Inclusion', this is referring to each of the three sub-arrays in the two-dimensional jagged array 'arrHighscores' which are all set up identically to show that the same results are obtained for all the sub-arrays.

### Driver Code

|  |
| --- |
| Private Sub TestSortHighscores()   Dim arrHighscores As Highscore()() = New Highscore(2)() {}  arrHighscores(0) = New Highscore(4) {}  arrHighscores(1) = New Highscore(4) {}  arrHighscores(2) = New Highscore(4) {}   *' Data Set 1*   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 50  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstInput1)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput1)   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 50  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstExpected1)   *' Data Set 2*   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 20  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 40  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 21  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 21  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 12  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 12  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstInput2)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput2)   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 20  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 40  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 21  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 21  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 12  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 12  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstExpected2)   *' Data Set 3*   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 50  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 10  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 40  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 20  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 30  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 31  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 41  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 11  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 51  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 21  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 52  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 22  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 32    DisplayHighscores(arrHighscores, lstInput3)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput3)   arrHighscores(0)(0).playerName = "Bob"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Damon"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Edvin"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Cheryl"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Alfred"  arrHighscores(0)(4).gameTime = 50  arrHighscores(1)(0).playerName = "Henry"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "Jacob"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Frank"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "George"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Isaac"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Monty"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Olivia"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Ludwig"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstExpected3)   *' Data Set 4*   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 50  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 40  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 20  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 10  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 51  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 41  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 21  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 11  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 52  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 42  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 22  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 12    DisplayHighscores(arrHighscores, lstInput4)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput4)   arrHighscores(0)(0).playerName = "Edvin"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Damon"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Bob"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Alfred"  arrHighscores(0)(4).gameTime = 50  arrHighscores(1)(0).playerName = "Jacob"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "Isaac"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "George"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Frank"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Olivia"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Nathan"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Ludwig"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Kevin"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstExpected4)   *' Data Set 5*   arrHighscores(0)(0).playerName = "Bob"  arrHighscores(0)(0).gameTime = 20  arrHighscores(0)(1).playerName = "Cheryl"  arrHighscores(0)(1).gameTime = 30  arrHighscores(0)(2).playerName = "Damon"  arrHighscores(0)(2).gameTime = 40  arrHighscores(0)(3).playerName = "Edvin"  arrHighscores(0)(3).gameTime = 50  arrHighscores(0)(4).playerName = "Alfred"  arrHighscores(0)(4).gameTime = 10  arrHighscores(1)(0).playerName = "George"  arrHighscores(1)(0).gameTime = 21  arrHighscores(1)(1).playerName = "Henry"  arrHighscores(1)(1).gameTime = 31  arrHighscores(1)(2).playerName = "Isaac"  arrHighscores(1)(2).gameTime = 41  arrHighscores(1)(3).playerName = "Jacob"  arrHighscores(1)(3).gameTime = 51  arrHighscores(1)(4).playerName = "Frank"  arrHighscores(1)(4).gameTime = 11  arrHighscores(2)(0).playerName = "Ludwig"  arrHighscores(2)(0).gameTime = 22  arrHighscores(2)(1).playerName = "Monty"  arrHighscores(2)(1).gameTime = 32  arrHighscores(2)(2).playerName = "Nathan"  arrHighscores(2)(2).gameTime = 42  arrHighscores(2)(3).playerName = "Olivia"  arrHighscores(2)(3).gameTime = 52  arrHighscores(2)(4).playerName = "Kevin"  arrHighscores(2)(4).gameTime = 12    DisplayHighscores(arrHighscores, lstInput5)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput5)   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 50  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 51  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 52   DisplayHighscores(arrHighscores, lstExpected5)   *' Data Set 6*   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 10  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 11  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 12   DisplayHighscores(arrHighscores, lstInput6)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput6)   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Edvin"  arrHighscores(0)(1).gameTime = 10  arrHighscores(0)(2).playerName = "Bob"  arrHighscores(0)(2).gameTime = 20  arrHighscores(0)(3).playerName = "Cheryl"  arrHighscores(0)(3).gameTime = 30  arrHighscores(0)(4).playerName = "Damon"  arrHighscores(0)(4).gameTime = 40  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "Jacob"  arrHighscores(1)(1).gameTime = 11  arrHighscores(1)(2).playerName = "George"  arrHighscores(1)(2).gameTime = 21  arrHighscores(1)(3).playerName = "Henry"  arrHighscores(1)(3).gameTime = 31  arrHighscores(1)(4).playerName = "Isaac"  arrHighscores(1)(4).gameTime = 41  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Olivia"  arrHighscores(2)(1).gameTime = 12  arrHighscores(2)(2).playerName = "Ludwig"  arrHighscores(2)(2).gameTime = 22  arrHighscores(2)(3).playerName = "Monty"  arrHighscores(2)(3).gameTime = 32  arrHighscores(2)(4).playerName = "Nathan"  arrHighscores(2)(4).gameTime = 42   DisplayHighscores(arrHighscores, lstExpected6)   *' Data Set 7*   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Bob"  arrHighscores(0)(1).gameTime = 20  arrHighscores(0)(2).playerName = "Cheryl"  arrHighscores(0)(2).gameTime = 30  arrHighscores(0)(3).playerName = "Damon"  arrHighscores(0)(3).gameTime = 40  arrHighscores(0)(4).playerName = "Edvin"  arrHighscores(0)(4).gameTime = 10  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "George"  arrHighscores(1)(1).gameTime = 21  arrHighscores(1)(2).playerName = "Henry"  arrHighscores(1)(2).gameTime = 31  arrHighscores(1)(3).playerName = "Isaac"  arrHighscores(1)(3).gameTime = 41  arrHighscores(1)(4).playerName = "Jacob"  arrHighscores(1)(4).gameTime = 11  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Ludwig"  arrHighscores(2)(1).gameTime = 22  arrHighscores(2)(2).playerName = "Monty"  arrHighscores(2)(2).gameTime = 32  arrHighscores(2)(3).playerName = "Nathan"  arrHighscores(2)(3).gameTime = 42  arrHighscores(2)(4).playerName = "Olivia"  arrHighscores(2)(4).gameTime = 12   DisplayHighscores(arrHighscores, lstInput7)   For i = 0 To arrHighscores.Length - 1 Step 1  SortHighscores(arrHighscores, i)  Next   DisplayHighscores(arrHighscores, lstOutput7)   arrHighscores(0)(0).playerName = "Alfred"  arrHighscores(0)(0).gameTime = 10  arrHighscores(0)(1).playerName = "Edvin"  arrHighscores(0)(1).gameTime = 10  arrHighscores(0)(2).playerName = "Bob"  arrHighscores(0)(2).gameTime = 20  arrHighscores(0)(3).playerName = "Cheryl"  arrHighscores(0)(3).gameTime = 30  arrHighscores(0)(4).playerName = "Damon"  arrHighscores(0)(4).gameTime = 40  arrHighscores(1)(0).playerName = "Frank"  arrHighscores(1)(0).gameTime = 11  arrHighscores(1)(1).playerName = "Jacob"  arrHighscores(1)(1).gameTime = 11  arrHighscores(1)(2).playerName = "George"  arrHighscores(1)(2).gameTime = 21  arrHighscores(1)(3).playerName = "Henry"  arrHighscores(1)(3).gameTime = 31  arrHighscores(1)(4).playerName = "Isaac"  arrHighscores(1)(4).gameTime = 41  arrHighscores(2)(0).playerName = "Kevin"  arrHighscores(2)(0).gameTime = 12  arrHighscores(2)(1).playerName = "Olivia"  arrHighscores(2)(1).gameTime = 12  arrHighscores(2)(2).playerName = "Ludwig"  arrHighscores(2)(2).gameTime = 22  arrHighscores(2)(3).playerName = "Monty"  arrHighscores(2)(3).gameTime = 32  arrHighscores(2)(4).playerName = "Nathan"  arrHighscores(2)(4).gameTime = 42   DisplayHighscores(arrHighscores, lstExpected7)   End Sub |

### 

### Test Result

### Discussion

All test data items returned their expected outputs. The subroutine is performing as expected. It should be noted that there is a defined ambiguity in the code - what happens when two or more users have an identical score? In that case, the behaviour is for the user who is already higher ranked (i.e. if one user was already 3rd and another user with that same time was in 5th) will stay in their location (so the user in 3rd will stay in 3rd) and the next user will follow behind them (so the user who achieved the same score, later will be in 4th), and this behaviour continues for all groups of users with the same scores. In effect, if any two users have the same score, they will remain in that order unless the highscores are manually modified (they will not be swapped in the bubble sort as the condition is '<' not '<=' or similar).

## 2 Function: ValidateSeed

### Code

|  |
| --- |
| Private Function ValidateSeed(enteredSeed As String) As Boolean   *' The pattern describes any hexadecimal digit (upper and lower case), 1-10 digits in length, making up the entirety of the string*  Dim pattern As Regex = New Regex("^[0-9a-fA-F]{1,10}$")  Return pattern.IsMatch(enteredSeed)  End Function |

#### Code Breakdown

Note that this code makes use of a regular expression, which is a pattern matching utility. It is similar to EBNF in the sense that it can specify sequence, repetition, and optional characters, but it has more flexibility as it can involve classes that are defined inline as well as numerical bounds for repetition (it's an extremely powerful pattern matching and string processing tool). The regular expression used to describe a valid seed is '^[0-9a-fA-F]{1,10}$'. This pattern matches a string whose entire contents are a single valid hexadecimal number. The breakdown is as follows:

|  |  |
| --- | --- |
| **Character/Token** | **Meaning/Purpose** |
| ^ | Match the beginning of the string (the following characters must be at the start of the string, there cannot be non-matching characters starting the string). This is to prevent matching substrings. |
| [0-9a-fA-F] | Defines a character class of the characters 0-9 (all decimal digits), a-f (lowercase letters a-f) and A-F (uppercase letters A-F). On its own, this would match a single character from the character class. |
| {1,10} | Modifies the preceding token to match between 1-10 (inclusive). In this case, any of the characters in the character class [0-9a-fA-F] can be matched from 1-10 times inclusive. |
| $ | Matches the end of the string (the preceding characters must be at the end of the string, there cannot be non-matching characters ending the string). This is to prevent matching substrings. |

So, the regular expression matches 1-10 characters (inclusive) that are the digits 0-9, and the uppercase and lowercase digits A-F; and the string must exist on its own (i.e., cannot be a substring). For more information, please consult the MDN Regular Expression cheat-sheet[1]:

[1]: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions/Cheatsheet>

### 

### Test Data

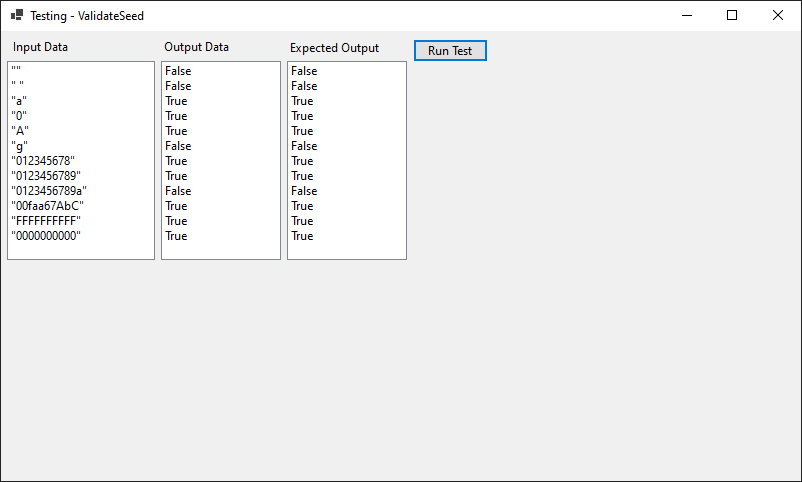
|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Variable Value** | **Expected Output** | **Reason for Inclusion** |
| enteredSeed | "" | False | Empty string, on lower bound of regex length specifier |
| enteredSeed | " " | False | Invalid character (whitespace) |
| enteredSeed | "a" | True | Single character (lowercase letter), above lower bound of regex length specifier |
| enteredSeed | "0" | True | Single character (digit), above lower bound of regex length specifier |
| enteredSeed | "A" | True | Single character (uppercase letter), above lower bound of regex length specifier |
| enteredSeed | "g" | False | Invalid character (non-hexadecimal) |
| enteredSeed | "012345678"  (9 chars) | True | Below upper bound of regex length specifier |
| enteredSeed | "0123456789"  (10 chars) | True | On upper bound of regex length specifier |
| enteredSeed | "0123456789a"  (11 chars) | False | Above upper bound of regex length specifier |
| enteredSeed | "00faa67AbC"  (10 chars) | True | Testing mix of digits, upper and lower case |
| enteredSeed | "FFFFFFFFFF"  (10 chars) | True | Testing maximum full length string value |
| enteredSeed | "0000000000"  (10 chars) | True | Testing minimum full length string value |

### 

### Driver Code

|  |
| --- |
| Private Sub TestValidateSeed()   Dim testDataArr As String() = {  "", " ", "a", "0", "A", "g", "012345678", "0123456789", "0123456789a", "00faa67AbC", "FFFFFFFFFF", "0000000000"  }   Dim expectedOutputArr As Boolean() = {  False, False, True, True, True, False, True, True, False, True, True, True  }   For i = 0 To testDataArr.Length - 1 Step 1  lstInputs.Items.Add("""" & testDataArr(i) & """")  lstOutput.Items.Add(ValidateSeed(testDataArr(i)))  lstExpected.Items.Add(expectedOutputArr(i))  Next  End Sub |

### Test Result



### Discussion

All test data items returned their expected outputs. The function is performing as expected.

## 3 Function: ValidateName

### Code

|  |
| --- |
| Private Function ValidateName(ByVal enteredName As String, ByRef playerName As String) As Boolean   Dim checkPassed As Boolean = True   *' Match the entered name against the pattern describing alphanumeric digits or underscores, 1-16 characters, for the entire string (i.e. not a substring, the whole string)*   Dim validNamePattern As Regex = New Regex("^[a-zA-Z0-9\_]{1,16}$")  If Not validNamePattern.IsMatch(enteredName) Then  checkPassed = False  Else  *' If the name is valid, assign it to the playerName variable and return True*  playerName = enteredName  End If   Return checkPassed  End Function |

#### Code Breakdown

Similar to ValidateSeed, this function makes use of a regular expression to validate the name that is entered. If you require more information regarding regular expressions, please see the 'Code Breakdown' section of ValidateSeed. The regex breakdown is as follows (with some more verbose explanations removed):

|  |  |
| --- | --- |
| **Character/Token** | **Meaning/Purpose** |
| ^ | Match the beginning of the string. This is to prevent matching substrings. |
| [a-zA-Z0-9\_] | Defines a character class of the characters a-z (all lowercase letters), A-Z (all uppercase letters), 0-9 (all decimal digits), and the underscore character. This will match a single occurrence of the class if not succeeded by a quantifier. Note that this class could have been made more concise with the use of the .NET specific \w character token, but I included the entire class for readability. |
| {1,16} | Modifies the preceding token to match between 1-16 (inclusive). In this case, any of the characters in the character class [a-zA-Z0-9\_] can be matched from 1-10 times inclusive. |
| $ | Matches the end of the string. This is to prevent matching substrings. |

### 

### Test Data

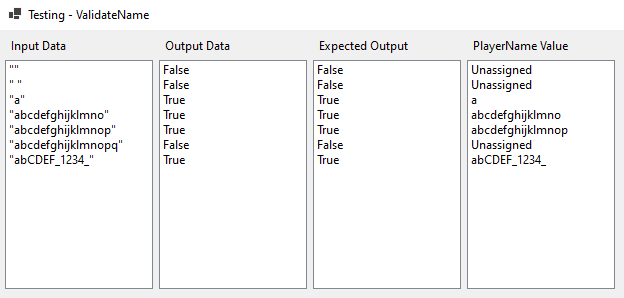
| **Variable Name** | **Variable Value** | **Expected Output** | **Reason for Inclusion** |
| --- | --- | --- | --- |
| enteredName | "" | False | Empty string, on lower bound of regex length specifier |
| enteredName | " " | False | Whitespace-only string |
| enteredName | "a" | True | Single character, above lower bound of regex length specifier |
| enteredName | "abcdefghijklmno" (15 chars) | True | Below upper bound of regex length specifier |
| enteredName | "abcdefghijklmnop" (16 chars) | True | On upper bound of regex length specifier |
| enteredName | "abcdefghijklmnopq" (17 chars) | False | Above upper bound of regex length specifier |
| enteredName | "abCDEF\_1234\_" | True | Test validity of digits, multiple cases, and underscores |

### 

### Driver Code

|  |
| --- |
| Private Sub TestValidateName()   Dim testDataArr As String() = {  "", " ", "a", "abcdefghijklmno", "abcdefghijklmnop", "abcdefghijklmnopq", "abcdef\_1234"  }   Dim expectedOutputArr As Boolean() = {  False, False, True, True, True, False, True, True, False, True, True, True  }   Dim playerName As String   For i = 0 To testDataArr.Length - 1 Step 1  playerName = "Unassigned"  lstInputs.Items.Add("""" & testDataArr(i) & """")  lstOutput.Items.Add(ValidateName(testDataArr(i), playerName))  lstExpected.Items.Add(expectedOutputArr(i))  lstPlayerName.Items.Add(playerName)  Next  End Sub |

### Test Result



### Discussion

All test data items returned their expected outputs. The function is performing as expected. Note that the auxiliary parameter 'playerName' was also tested and its values after each call were as expected (with the default value 'Unassigned' not being in the test data set to prevent any overlapping values / undefined logic, and to provide a clear response if it was unmodified).

## 

## Testing on different specifications

### Description

This testing was performed to analyse the impact of differing computer specifications on the program's performance. Factors such as responsiveness and delays, screen lag and pixelation issues, were all evaluated whilst playing the game, as well as the overall computer load during the gameplay. The tests were performed on the development machine to provide a best-case scenario (i.e., there should be no issues with the game or noticeable load whatsoever), and a very old laptop running the same software to provide a worst-case scenario (i.e., all issues that could occur regarding performance would be noticed in this case). The specifications for the two machines are as follows (both were running the latest build of Windows 10):

High-end machine (best-case game runtime scenario):

* Apple MacBook Pro 2019 16"
* Intel Core i7 9750H 6-core CPU running at 3.98GHz boost clock (2.60GHz base clock)
* 16GB of RAM running at 2667 MHz (specification listing)
* AMD Radeon Pro 5500M Discrete GPU with 8GB of VRAM
* Internal display running at 3072x1920 resolution, 60Hz refresh rate

Low-end machine (worst-case game runtime scenario):

* Apple MacBook Pro 2013 13"
* Intel Core i5 4288U 2-core CPU running at 3.10GHz boost clock (2.60GHz base clock)
* 8GB of RAM running at 1600MHz (specification listing)
* Intel Iris 5100 Integrated Graphics with 128MB of Dedicated GPU Memory
* Internal display running at 2560x1600 resolution, 60Hz refresh rate

### Evidence and Results

|  |  |
| --- | --- |
| Trial 1 | Trial 2 |
|  |  |
| The game ran perfectly smoothly with 0 delays, as was expected. There was no latency when changing forms or maze sizes, and the game rendering was within a single frame (so less than 1/60th of a second). This was an anticipated result since there had been no delays experienced during development (and during the development process the game was run perfectly with Debug mode activated, which adds significant latency - so it was to be expected that running with Debug mode disabled would be faster to the point of instantaneous updates, which was the case). | The game ran smoothly, with no delays during the gameplay itself (the frmGame screen). Extremely slight delays were experienced on frmMain and frmHighscores when switching the displayed difficulties - this is likely due to the rendering of the Flat appearance of the custom ButtonEllipse class; this was not implemented in .NET by default and thus lacks all the optimisations that .NET includes for standard control re-rendering. The additional compute power required to manage these rendering tasks is likely what added to the additional delay - although, it was timed to be between 0.07-0.09 seconds using a 240FPS recording - this is not sufficient to require a notice to the user (a progress bar, for example), but is noticeable if one is expecting instantaneous feedback (as was experienced on the high-end machine). There were no additional delays within the program, and the startup time was near instant also (as cached local image files were used, and the display resolution was slightly lower than the high-end machine, less video output had to be written). |

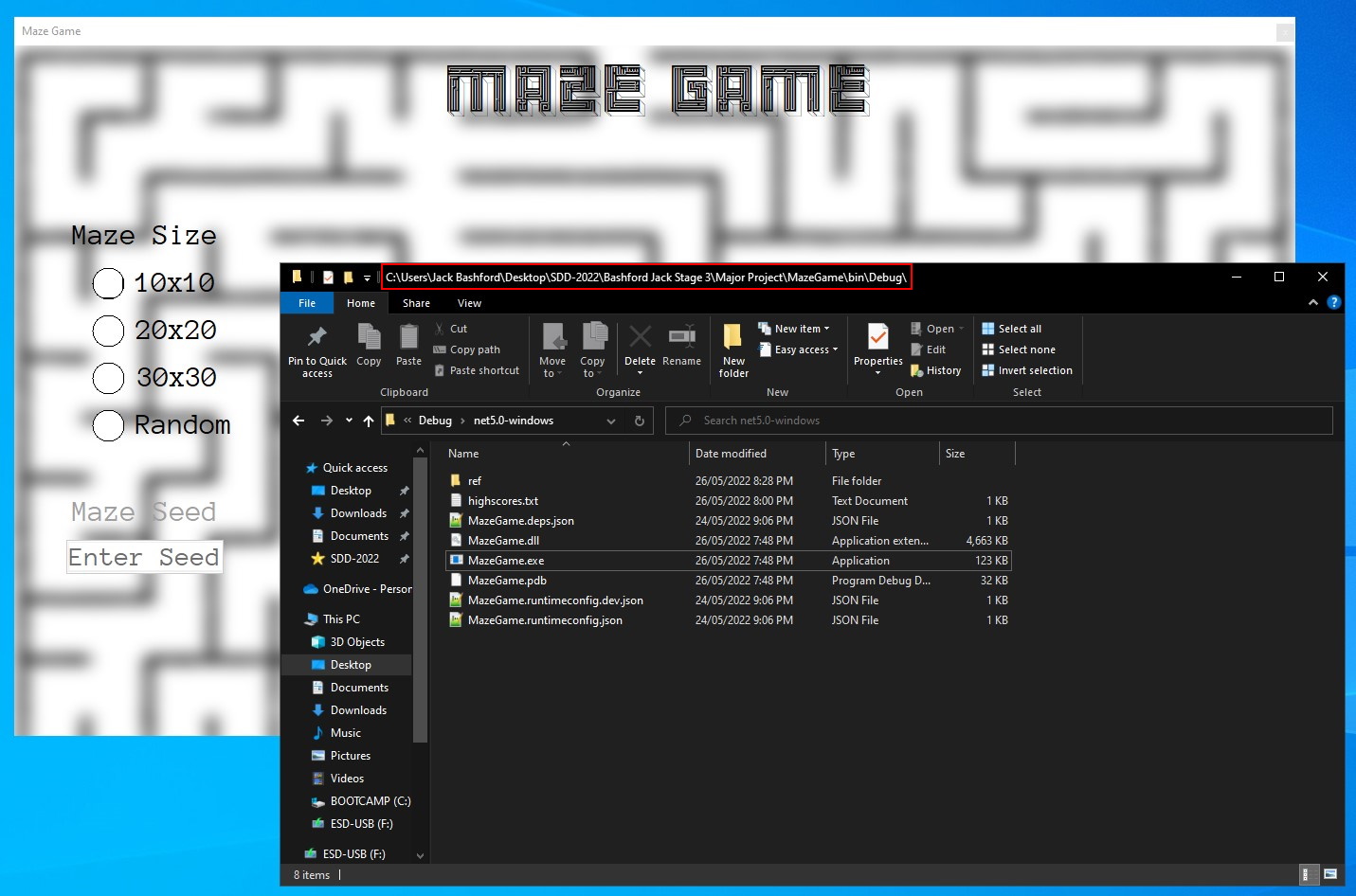
## Testing with different install locations

### Description

This test will examine the impact of installing and running the game from two different locations with varying data bus speeds (both tests will be performed on the high-end machine described in 'Testing on different specifications'). The two locations will be:

1. The Downloads folder of the local drive of the high-end machine. This is technically an external Samsung T7 SSD, but testing found less than 1.2% difference in read/write speeds between this drive and the internal SSD of the MacBook Pro. This drive represents a best-case scenario (in terms of read/write speeds) and is connected via USB-C Gen 3.2.
2. The root of an external 8GB USB that has had several terabytes of total data passthrough. Due to the MacBook Pro 16" 2019 having extremely limited I/O (a total of 4 USB-C Thunderbolt 3 ports), an adapter must be used - and it is connected via USB-A Gen 2.0 (via an Apple brand USB-C to USB-A adapter).

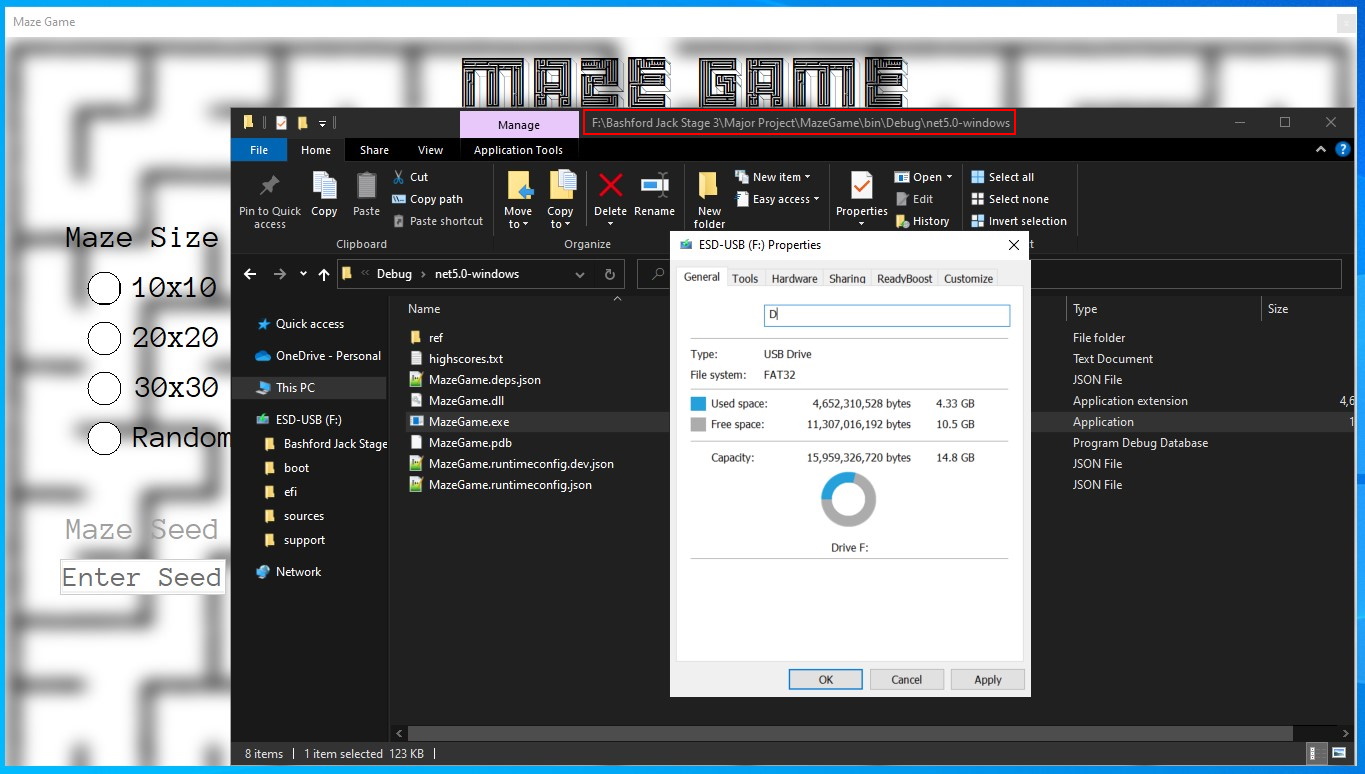
### Evidence 1



### Results 1

As expected, with the game installed locally on the boot drive of the computer, it ran entirely as smoothly as expected, with no lag / long pauses during the file write calls (which occur when the game starts and frmMain is shown, and once you win the game and frmHighscores is shown).

### Evidence 2



### Results 2

Given the extremely small quantity of data that was required to be written / read by this program, I did not expect any significant delays in these areas (and there were no more delays when showing frmHighscores, as expected). However, the game startup initially was slower than I had anticipated - I believe this is due to the game having to pull each of the required DLLs and other execution-required packages from the USB into the main storage disk temporarily and then into RAM, which adds additional time (as the USB-A Gen 2.0 bus, through an adapter, is far slower than a USB-C Gen 3.2 bus with no adapter). This added between 0.2-0.3s of startup time upon a timed startup of the application on both local and external installation locations. However, it should be noted that this is negligible, and I only noticed it because I was searching for it - testing with other individuals revealed that they could not tell the difference between a local/external startup.

## Changes required to program

As noted above, there was not a noticeable delay in the startup times (less than 0.3 seconds in 90% of the time, upon testing) when comparing between a local and external installation, and this was mainly due to requirements for caching - if the game is already in RAM or a cache on the internal/boot drive, then this would dramatically speed up the startup. The file writing segments of the game had absolutely no noticeable/timed latency difference, with both being effectively instantaneous in response time. This indicates that the difference in installation locations only affects the startup time of the application and not the file handling performance, therefore there are no changes required to the program.

# Evaluation

## Design Specifications

Note that these design specifications & screens are taken directly from the Stage 1 & 2 document.

### User’s Perspective

| **Design Specification** | **Discussion** | **Specification Met?** |
| --- | --- | --- |
| The game will have three primary difficulties, with varying maze sizes (standard sizes of 10x10, 20x20, and 30x30, with a random choice of those three available as well). | This specification was one of the most closely worked-with during the game's development - and it was one of the easiest specifications to meet, as well as one that was required to make sure that the screen designs were appropriately achieved. |  |
| A main screen with access to the highscores, instructions, and gameplay screens will be available to view upon the initial launch of the program. | This was made entirely as listed - it should also be noted that this included the entries for the player's name, desired difficulty, and maze seed (if any). |  |
| An ending screen with access to the highscores and main screens will be available to view upon the conclusion of the game. This will display the time taken through the maze, and ask whether the player would like to view highscores, play again, or exit the game. | This was initially accomplished but I was unable to implement a 'play again' feature, due to graphics scaling issues (the primary problem was that the graphics scaling persisted between game sessions, even when resets were attempted). |  |
| A highscores screen will be available to view from the main screen and the ending screen. This will display the fastest times through a maze of the same size if playing with any of the three standard sizes (10x10, 20x20, 30x30). | This was accomplished early on and was easy to achieve since it simply involved reading from the highscores file and writing the information out to the screen. It was implemented with no changes from the screen designs. |  |
| An instructions screen will be available to view from the main screen. This will display the instructions for playing the game. | This was very easy to implement, and only required changes in text content from the original plans (note that these are not included in the design specification, and thus changing the content does not mean the specification was not met). |  |
| A gameplay screen will be available to view from the main screen. This will display the maze, user's position, and elapsed time throughout the game. | This is the main component of the game and was implemented relatively easily - the only problems being the drawing of the maze screen faced lots of issues during development, but these were resolved after significant testing and debugging. |  |
| The game's timer starts as soon as the maze is created and the user is able to interact with the game, and the timer ends as soon as the player reaches the finish cell of the maze. | This was a relatively small and simple component of frmGame and was quite quick and easy to implement with no major issues. |  |
| The user can move their player through the maze with keyboard or mouse shortcuts. They will only be able to move into open cells, not cells blocked by walls or the edge of the maze. | The keyboard was the hardest component to work on here, as I had to ensure that the text boxes weren't being focused when the arrow keys were pressed. The mouse shortcuts were the initial control method that I implemented before the keyboard shortcuts. |  |
| The user may choose to prematurely exit the maze, which will not allow the user to achieve a high score ranking position. | This was not implemented as a game feature, it's only possible to do so if the user closes the game (with the Close button in the title bar, or with Alt-F4). |  |
| At the conclusion of each game, the user will be taken to the ending screen, and from there they will be able to view highscores and/or start a new game. | The only feature that was not possible to implement (as described above in the 'ending screen' specification) was the re-playing of the game (i.e., playing another game from within the same game session). Since this feature was included in the specification, this means that the specification as not met entirely, but it should be noted that all of the other components of this specification were met in their entirety. |  |

Developer’s Perspective

| **Design Specification** | **Discussion** | **Spec Met?** |
| --- | --- | --- |
| The maze will be stored as a 2D array arrGameBoard, consisting of cells represented as integers.  Each cell will contain a binary number (0000-1111) which is the mask of the walls that are open on that cell (i.e., 0010 would mean the southern wall is open only). | This specification was extremely critical and was one of the 'axioms' - the types that should not be modified to prevent massive changes to other parts of the program. The example given in the specification (0010 or 2 indicating the southern wall being open) is not what the final implementation used (it was 0100 or 4 in the final implementation) however as this is purely an example, the specification was still met. |  |
| The maze will be generated using a randomised algorithm that is based upon a seed (this seed can be used to predictively randomise the Randomize() function in VB.NET).  This can be used to regenerate the same maze from a certain seed if the user wishes to re-play that maze. Note it will only work for mazes of the same dimensions, i.e., a smaller portion of a maze cannot be generated with a seed of a larger maze.  The algorithm (recursive backtracking) creates a simply connected maze (i.e., it is made from branching passages and contains no loops) which is guaranteed to have a solution. | This specification was met and was implemented/tested in another language (Python) prior to its implementation in VB.NET. It is worth noting that significant issues were met during implementation due to the equivalent syntax for && (logical AND) and & (bitwise AND) in VB (both are 'And'), but after thorough debugging these issues were resolved.  The Randomize() function was not used in the end due to seed issues (based on whether the seed had to be generated or not, it would change the maze produced by the same seed - not the desired behaviour) - so instead the new Random class was used. The maze algorithm also successfully produced a simply connected maze. |  |
| The player's position will be represented by a tuple playerPosition (Integer, Integer) of the x-y coordinates relative to the starting position / start cell. Each movement will change the relevant component direction's value within that tuple by a value of 1 or -1. | The player's position was not represented by a tuple 'playerPosition' but instead by a tuple named 'coords'. The functionality of the tuple was identical to the design specifications, but the name differs to what was outlined in the specification, so the specification was not met. (It is accessed via coords.X and coords.Y, and these increase right and down respectively) |  |
| The game screen will be drawn using the Windows GDI+ graphics engine; this will allow for a range of shapes to be drawn, including pixel-by-pixel drawings.  Each combination of walls and open sides will be drawn following the bitmask rules for each cell (this will only have to happen once at the start of the game, and will be a linear operation with a worst-case runtime complexity of O(4n) which is equivalent to O(n). | This specification was met as it is, once again, a crucial component of the final game and was used as an assumption. The game uses the Windows GDI+ graphics engine successfully with minimal calls (to minimise lag and on-screen latency, as well as any flickering/flow-down effects that sometimes occur with graphics-intensive processes). The walls were drawn once at the start of the game, and this was a linear operation as expected. In addition, the start and end screens were drawn (as well as some fill-in blocks at the joints between walls, to make the viewing easier for the user). |  |
| When the game starts, the maze will be generated and drawn, the user will be drawn at the start cell, and the timer will begin as soon as the user is able to move. The timer will stop as soon as the user enters the final / winning square. | This specification was met and is more of a general outline of the frmGame processing order but is still highly relevant. This was the mainline, in effect, of the frmGame event handling, and it was met with very little issues. |  |
| Upon each movement command, the game will either move the user in the requested direction if it's possible to do so, or nothing will occur as there is a wall or maze boundary. | To achieve this specification, all that was required was to make working movement controls, and that was successfully done. Note that the same actions are performed whether a button is clicked, or a keystroke is detected (they both call the same direction handling subroutine). |  |
| The game's highscores will be represented within the game as an array of records for each maze size (it will be fetched upon loading of the highscores form, to avoid unnecessary lag and system resource usage when starting playing the game).  The record recGameResult will be used to represent the result of a game, containing the player's name (String), the maze dimensions (Integer), the elapsed time in milliseconds (Integer), and the maze's seed (Integer). This is a 'temporary' record (i.e. a maximum of only one data item of this record type will exist at any point in time) as it's either discarded if not a highscore time, or its data is copied into a new recHighscore to be stored in the highscores.txt file and the recGameResult value is discarded. As such, it does not appear in any system models because it will be entirely contained within one subroutine (checkIfHighscore). | The first part of this specification was partially met (an array of arrays of records, one array of records for each maze size) - the main issue was that the highscores were loaded when frmMain loaded, as frmHighscores needed to be shown multiple times and the load event only fires once (and the show event did not fire the second time it was shown for unknown reasons).  recGameResult was also never used, instead all highscores were stored as recHighscore values, and the first 5 values for each maze size in arrHighscores were taken via truncation (that is, the values were appended, the array was sorted, and then if there were any values beyond the first 5 for each maze size, they were discarded). |  |
| The project as a whole will be constructed with the top-down development method, using the structured software development approach. This will ensure that the resulting program is sufficiently robust with no flaws upon usage, but is also clearly structured in a hierarchical manner within the source code.  The source code will contain both internal and intrinsic documentation to allow for maintenance and optimisation of the program or any of its submodules | This was almost entirely achieved, however this did not consider the pre-development of the GenerateMaze subroutine which was a lower-level subroutine generated before its higher-level calling subroutine. Nearly all other subroutines were implemented in a top-down manner starting with the mainline, but the GenerateMaze was developed and tested in complete isolation, and implemented without a stub, meaning this specification was not met. |  |

### 

### Screens

#### frmMain

|  |  |
| --- | --- |
| **Final Game** | **Original Design** |
|  |  |
| There was no real change between these two forms in terms of on-screen appearance - the most drastic change was the smaller font size for everything except the title in the final design (it's evident that the buttons, the maze size and maze seed, and textboxes, all have smaller font sizes than the initial design). No additional elements were added nor were elements removed from the screen. The background image was also slightly less blurred in the game, which did cause some extremely minor readability issues in certain places (the word 'Maze' in 'Maze Size' for instance). There are no functionality changes between the two forms. | |

#### 

#### frmInstructions

|  |  |
| --- | --- |
| **Final Game** | **Original Design** |
|  |  |
| The only notable change in this form, apart from the noted decreased text size and less blurred background, was the addition of a small amount of text (the third paragraph, noting the maze seed and its format) and some rearrangement of other text. All other parts of this form remained unchanged, and there are no functionality changes between the two forms. | |

#### frmGame

|  |  |
| --- | --- |
| **Final Game** | **Original Design** |
|  |  |
| There are two small changes in this form. The first relates to the controls - the username and timer were moved slightly down so they are now on the same line as the first row of the game, as opposed to alongside the title. (The noted background and font size changes are also in effect on this form). The other change relates to the content of the game panel - these changes, all itemised, are:   * The internal maze walls are doubly thick compared to the external walls (compared to the original design, where the maze walls were all the same thickness). This was due to the lowest common multiple of the number of dynamic pixels required to display the mazes in the original configuration exceeded 777000, and thus this was infeasible. Instead, the choice was made to repeat the internal walls to display all three possible maze sizes in a total of 600 pixels of space. * The starting and ending squares have different designs - in the original design, there was no design on the starting square and a chequered flag on the ending square. In the final design, a green square was used for the starting square and a golden square was used for the ending square. * The player's avatar was represented by a full-size CadetBlue square in the game, as opposed to a 1-pixel Blue square in the designs. * It should also be noted that the game panel is larger in the final game than in the original design.   There are no functionality changes between the two forms. | |

#### frmGameOver

|  |  |
| --- | --- |
| **Final Game** | **Original Design** |
|  |  |
| The only interface changes between these two forms are the noted background colour changes and font size changes, as well as the lack of a boundary box around the maze seed. The only functionality change is that the 'Back to Start' button is now 'Quit Game' instead. | |

#### 

#### frmHighscores

|  |  |
| --- | --- |
| **Final Game** | **Original Design** |
|  |  |
| The only changes between these two forms are the noted background colour changes and font size changes. There are no functionality changes between the two forms. | |

#### frmHelp

|  |  |
| --- | --- |
| **Final Game** | **Original Design** |
|  | This form was not included in the original screen designs. |
| This form was not included in the original screen designs. It was included to have an in-game troubleshooting guide (it's accessible when pressing F1 on any form). It is very similar in design to frmInstructions, with the only difference being the title and the contents of the screen. The difference between this form and the original design cannot be evaluated since no original design was provided. | |

## User Evaluation

### Questions

1. How much experience do you have with using computers? (i.e., file handling, troubleshooting, installation, etc.)
2. How much experience do you have solving mazes/puzzles?
3. Did you have any issues installing the game or any of its assets (fonts)?
   * If you answered 'Yes' above, please describe the issue/s you faced.
4. Were there any features/elements of the game that inconvenienced or annoyed you?
   * If you answered 'Yes' above, please describe the issues you faced.
5. How useful were the Instructions and Help forms for in-game assistance?
   * What improvements would you suggest to the Instructions and Help forms?
6. How useful was the User Guide for detailed information about the game?
   * What improvements would you suggest to the User Guide?
7. How useful was the User Tutorial for learning how to play the game?
   * What improvements would you suggest to the User Tutorial?
8. Overall, how would you rate your enjoyment of the game?

### Evidence of survey

|  |
| --- |
|  |

### Overall survey summary

In general, the game was very well-received. The user guide, tutorial, and in-game forms were useful for a majority of the survey respondents in learning how to play the game, and there were not many issues related to the game's installation or documentation. Most users found the lack of a 'Play Again' button disappointing and expressed that they would have preferred to see more information regarding the maze seed. There was also evidence that not all users had read through all the documentation in its entirety, so perhaps additional emphasis on certain parts of the documentation (through colouring text on the in-game forms, for instance) would be valuable modifications to make in the future.

### 

### Individual Response Summary

#### Question 1: How much experience do you have with using computers? (i.e., file handling, troubleshooting, installation, etc.)

All of the respondents have above average computer skills – this slightly limits the applicability and diversity of the survey results but also ensures that more high-level feedback will be provided.

#### Question 2: How much experience do you have solving mazes/puzzles?

All of the respondents have at least some experience solving, this means that the respondents will be able to complete the game (hopefully) and provide feedback about the game in a coherent manner, rather than being unable to complete the game.

#### Question 3: Did you have any issues installing the game or any of its assets (fonts)?

The majority of the respondents were able to install the game with no issues. However, two respondents faced issues - one had a .NET version that was higher than the required version (5.0) and downgrading it is very difficult to do, and one did not install the fonts correctly (which was detailed in the user guide).

#### Question 4: Were there any features/elements of the game that inconvenienced or annoyed you?

There were several respondents who had feedback regarding game features that were inconvenient, annoying, or otherwise less than perfect. Itemised and summarised, they are:

* The lack of the 'Play Again' functionality in the game
* The lack of a 'decision point' movement style (for larger mazes) - when moving through the maze, instead of moving one square in a given direction, the player would instead be moved to the next junction/turnoff point in that given direction. This would be easy to do for straight paths, but very difficult to implement for a single path with lots of corners as then a path-following algorithm would be required at every stage, decreasing efficiency massively.
* The implicit username / seed validity checking (i.e., you couldn't click the 'Play' button without a valid username and seed) was confusing to users at first (however they would have noted the valid formats were detailed in the User Guide and in-game assistance).
* Several smaller issues regarding the UI and lack of tab-ability to the 'Play' button (due to its being disabled), as well as the appearance of certain textboxes (looking like buttons).

Several of these issues are preferential and entirely cosmetic, with only three ('Play Again' functionality, decision point movement, and seed/validity checking explicitly) requiring functional modifications to the code (and two of those are extremely simple to do; decision point movement is extremely difficult to accomplish, and would best be done during maze generation to note points for each cell that would be the final position that the user would end up at).

#### Question 5: How useful were the Instructions and Help forms for in-game assistance, and what improvements would you suggest?

The majority of the respondents had positive feedback for the in-game assistance, with the common points of improvement being:

* The help form was difficult to access (it is accessed via F1 as detailed in the User Guide)
* Split up the help/instructions forms - one part of each should relate to general gameplay, and there should be a separate form relating to usage of the seed functionality
* Formatting could have been improved as the explanations were quite verbose.

Overall, feedback was quite positive for this aspect of the documentation, with multiple comments that it required no improvements at all.

#### Question 6: How useful was the User Guide for detailed information about the game, and what improvements would you suggest?

The majority of the respondents had very positive feedback for the User Guide, with the common points of improvement being:

* Example seeds and screenshots of frmGame should be provided to illustrate specifically how the seed functionality works from a user's perspective.
* Including annotated diagrams with callouts, rather than text and images being separate, may have been clearer to understand.

Overall, feedback was overwhelmingly positive with only minor, primarily cosmetic changes being suggested.

#### Question 7: How useful was the User Tutorial for learning how to play the game, and what improvements would you suggest?

The majority of the respondents had extremely positive feedback for the User Tutorial, with the common points of improvement being:

* A selection of a different difficulty rather than just 10x10 would have shown the differences between the dimensions (as 30x30 is quite a large maze size compared to 10x10, with 9 times more cells to possibly traverse).
* A full level of the maze game would have allowed users to get more experience working with the controls and understand the objective more clearly (although I think it is clear that the objective is the golden square, others may not feel this way and may require a more thorough / verbose introduction to the gameplay mechanics).
* More information about the game seed (i.e., entering different / the same seeds and observing the changes) as well as more emphasis on the limitations of the seed values.

Overall, feedback was overwhelmingly positive with minor points being selected that would not have added much to the tutorial (since they were expansion-style points, with most of the suggested additional information already being available in the game and documentation). Adding a full level, or a different difficulty, would have increased the required complexity of the tutorial significantly.

#### Question 8: Overall, how would you rate your enjoyment of the game?

The feedback of the game was overall quite high, with the main points being provided over the survey being:

* There are several areas where users have preferences different to the game's implementation in certain scenarios - for instance, the decision point movement style.
* Many other areas focused on the addition of information that's not required but may be useful or beneficial cosmetically or for less experienced users.
* The game as a whole was extremely well-received, with only minor points being noted that would warrant changes to the game (rather than additional information being provided).