TABLE OF CONTENTS

DEFINING + UNDERSTANDING

PROBLEM STATEMENT

TASK

My task is to develop a number guessing game for *'Gametech'*, to be included in a package called *"PC Fun Zone 2015"*. This package will comprise multiple small games, and will be sold by *Gametech*.

CLIENT & REQUIREMENTS

The requirements for the project have been determined by Gametech:

The number guessing game must:

- Have an appropriate name and logo
- Have a consistent GUI and colour scheme
- Include a login screen that accepts a user name
- Generate a new random number for each game
- Include multiple difficulty levels:

	Normal	Epic	Legendary
Max number	10	100	1000
Number of guesses	3	5	9

- Provide appropriate user feedback after each guess such as "Too high", "Too low", "Number higher than Max" and "Please enter a number".
- Include a scoring system that ranks players. Less guesses means a higher score.
- The scoring system must keep track of player's username, score and difficulty level.

Possible extensions:

- Add a timing feature that tracks the number of seconds taken to play each game. Use this to calculate scores or possibly display in hall of fame.
- Add a timed mode, where time is limited instead of the number of guesses.
- Add ability to locate the most successful player for each level in the hall of fame.

RESOURCES AVAILABLE

I will be using various software applications/online services in order to create my number guessing application. These include:

- Microsoft Visual Studio 2010 – for implementing all of the working code and final solution
- Microsoft Office Suite – Microsoft Word for portfolio work, Microsoft Excel for Gantt chart and Microsoft PowerPoint for GUI mockups.

Dia – for flow charts, context diagrams, data flow diagrams and structure charts.

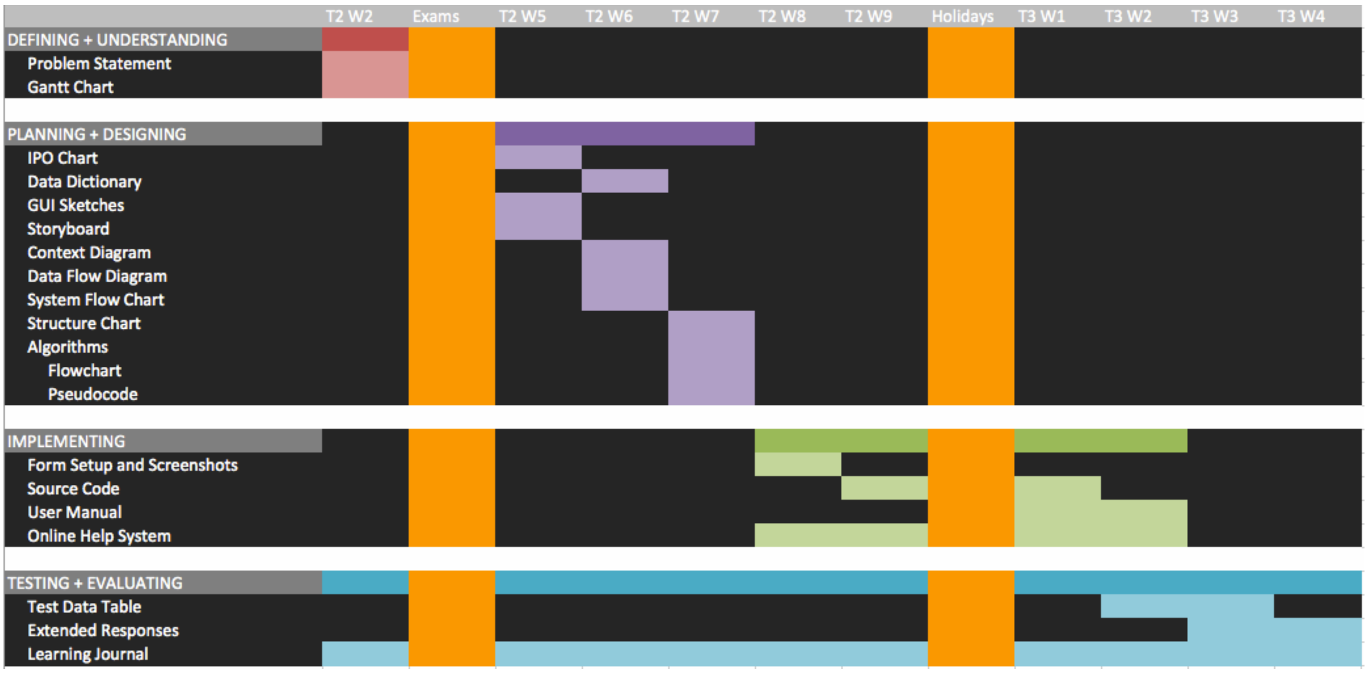
- Adobe Illustrator and Photoshop – for creating original graphics to use in my application to add to the visual appeal and overall user experience.
- Google Blogger – for keeping track of learning journal entries and logging new ideas and concepts as I come up with them

The hardware that I will have access to in order to create my application includes:

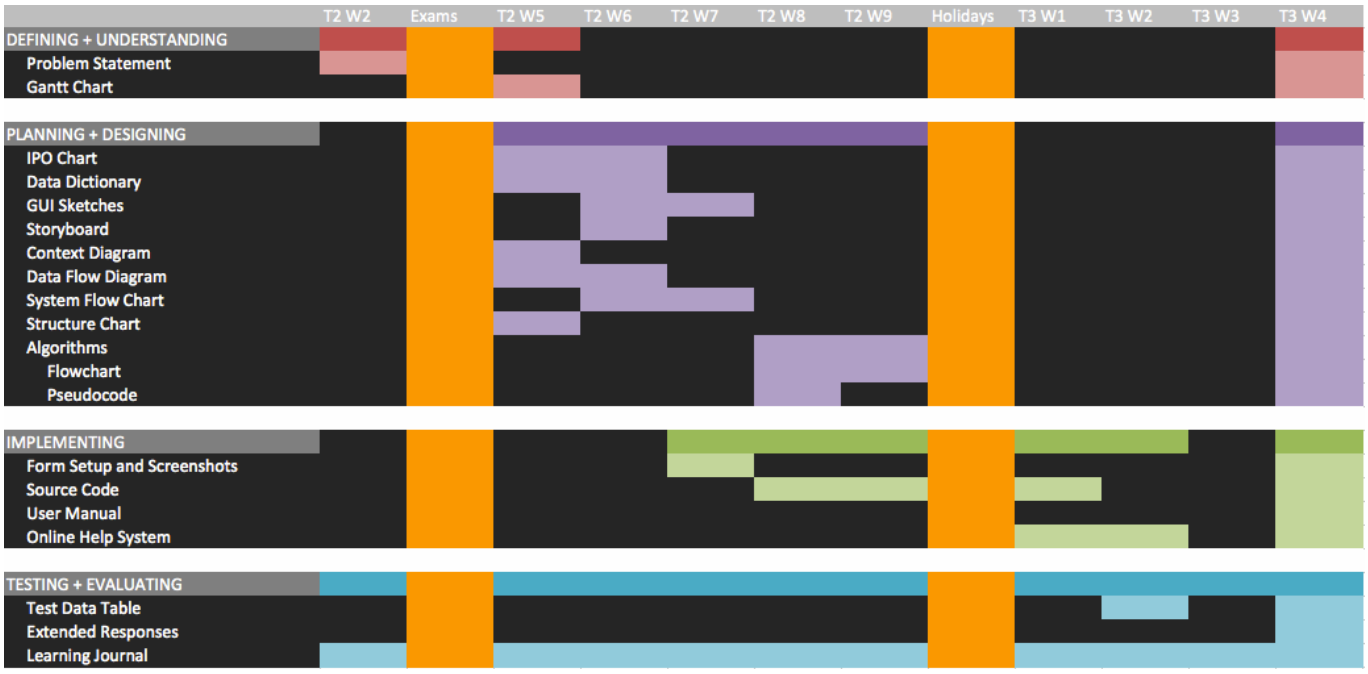
- A desktop PC running Windows 7 Professional. This computer has 8GB of RAM and has an Intel Core i5 CPU
- Along with the PC I will be using standard accessories such as a keyboard, mouse, headphones, microphone, printer, scanner, USB, etc.

GANTT CHARTS

ORIGINAL


T2 W2
Exams
T2 W5
T2 W6
T2 W7
T2 W8
T2 W9
Holidays
T3 W1
T3 W2
T3 W3
T3 W4
DEFINING + UNDERSTANDING
Problem Statement
Gantt Chart
PLANNING + DESIGNING
IPO Chart
Data Dictionary
GUI Sketches
Storyboard
Context Diagram
Data Flow Diagram
System Flow Chart
Structure Chart
Algorithms
Flowchart
Pseudocode
IMPLEMENTING
Form Setup and Screenshots
Source Code
User Manual
Online Help System
TESTING + EVALUATING
Test Data Table
Extended Responses
Learning Journal

REVISED


T2 W2
Exams
T2 W5
T2 W6
T2 W7
T2 W8
T2 W9
Holidays
T3 W1
T3 W2
T3 W3
T3 W4
DEFINING + UNDERSTANDING
Problem Statement
Gantt Chart
PLANNING + DESIGNING
IPO Chart
Data Dictionary
GUI Sketches
Storyboard
Context Diagram
Data Flow Diagram
System Flow Chart
Structure Chart
Algorithms
Flowchart
Pseudocode
IMPLEMENTING
Form Setup and Screenshots
Source Code
User Manual
Online Help System
TESTING + EVALUATING
Test Data Table
Extended Responses
Learning Journal

PLANNING + DESIGNING

IPO CHART

INPUT	PROCESSING	OUTPUT
Player Name	The player's name, stored as a string by the system. Passed between various forms for use in labels/hall of fame.	Name displayed on Main menu form Names displayed as part of a record in the hall of fame.
Game Difficulty	Game difficulty determines the max number of guesses, as well as the upper limit of the random number to be generated.	N/A
Guess	Compare guess with the randomly generated number If correct, generate score and load hall of fame. Also write game results to file (comprises Game ID, Player Name, Game Difficulty and Score.)	Display higher/lower graphic and update user feedback label. Load hall of fame
Hall of Fame search criteria	Filter all game results stored in text file by level difficulty. Sort the filtered results into decreasing order by score (note that only the top ten results need to be found.)	Display filtered/sorted results in the Hall of Fame table.

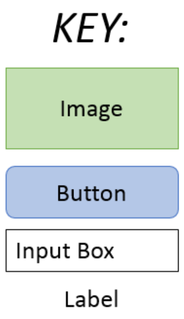
DATA DICTIONARY

NAME	DATA TYPE	SCOPE	DESCRIPTION	EXAMPLE	VALIDATION	FORMAT
			PUBLIC MODULE			
playerName	String	public	The player's name	"Alex"	<= 16 characters	
levelDifficulty	String	public	The difficulty level of the next game to be played, determined by user.	"Normal"		
			GAME FORM			
maxGuesses	Integer	private	Maximum number of guesses, determined by levelDifficulty	3		
maxNum	Integer	private	Maximum random number that could be generated by the program. Determined by levelDifficulty	10		
randomNum	Integer	private	Random number produced by program	7		
playerGuess	Integer	private	Guess entered by player	6	Integer between 1 and maxNum Can't contain letters or symbols	
guessesRemaining	Integer	private	Guesses remaining for the current game	1		
score	Integer	private	Score generated by game			
temp	Integer	private	Temporary variable for storing the randomly generated number.	37		
guessOK	Boolean	private	Flag that is set to true when the guess has been validated.	true		
			HALL OF FAME			
arrRecordsToDisplay	Array (score Record)	private	List of data from previously won games, filtered/sorted appropriately.	N/A – this data type is an array of records		
firstIndex	Integer	private	The first index to start the sort from	0		

lastIndex	Integer	private	The last index in the array to be sorted	42		
endUnsortedIndex	Integer	private	The number of items in the array that have already been sorted	5		
currentIndex	Integer	private	The index of the array element that is currently being sorted	8		
maxItem	Score Record	private	The scoreRecord with the maximum score value	1 “Normal” “Alex” 8		
maxIndex	Integer	private	The index of the scoreRecord with the maximum score value.	7		
temp	Score Record	private	Temporary variable for holding data while elements in the array are reshuffled.	1 “Normal” “Alex” 8		
arrToReturn	Score Record	private	The array to be returned after processing in the FilterScoresBy CurrentPlayer() or FilterScoresBy Difficulty() subroutines	N/A – this data type is an array of records		
newID	Integer	private	The next ID in the array arrToReturn that we want to assign to.	4		
READ WRITE DATA						
scoreRecord	Structure/ Record	public	Record of each win to be used in hall of fame	1 “Normal” “Alex” 8		ID (Int) Difficulty (String) Name (String) Score (Int)
arrScores	Array (score Record)	public	List of all data from previously won games.	N/A – this data type is an array of records		
filePath	String	public	The directory of the project.	\\net\ CD-roms\ Mirrington\ 11SDD Major Project		
oRead	System.IO .Stream Reader	private	Method for sequentially accessing the contents of text	N/A		

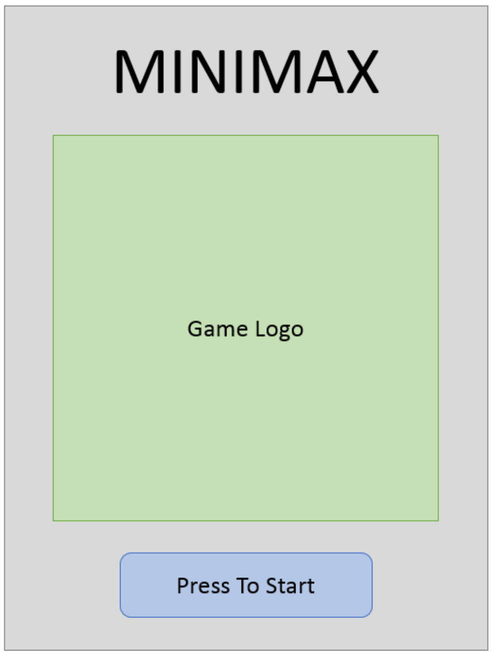
			files.			
lineIn	String	private	The line read by oRead that is then processed.	1|Normal|Alex|8		
tmp	Object	private	Temporary variable for storing data while it is processed and placed into arrScores.	N/A		
x	Integer	private	The previous last ID in arrScores	4		
NextID	Integer	private	The next ID in the array arrScores that we want to assign to.	5		
sep	Char	private	The separator character between elements in the text file.	|		
writeLine	String	private	The string to be written to file	1|Normal|Alex|8		

GUI SKETCHES



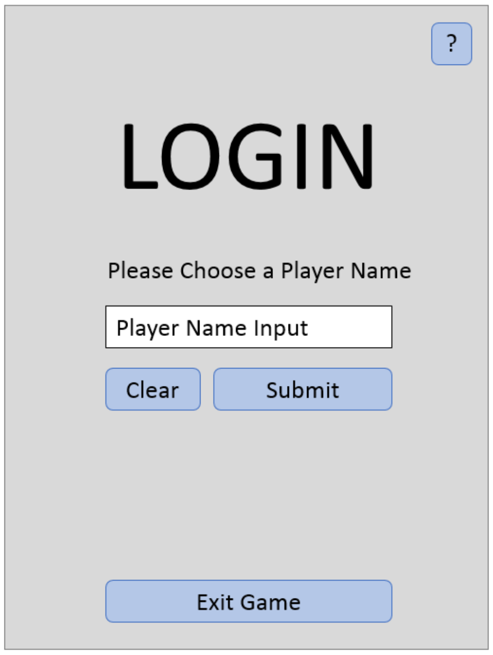
SPLASH SCREEN

450x600



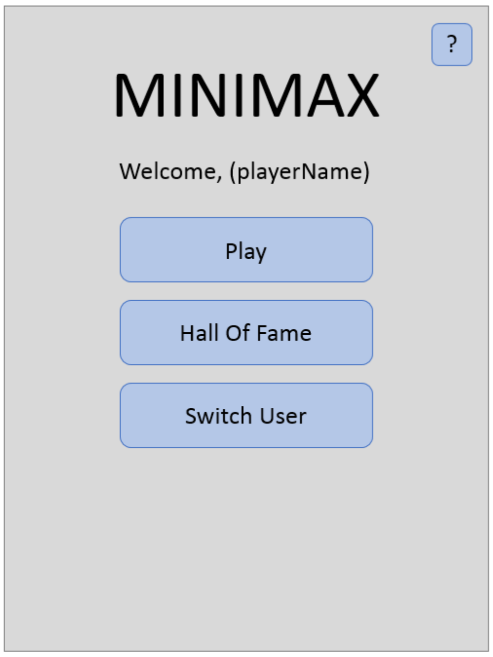
LOGIN SCREEN

450x600



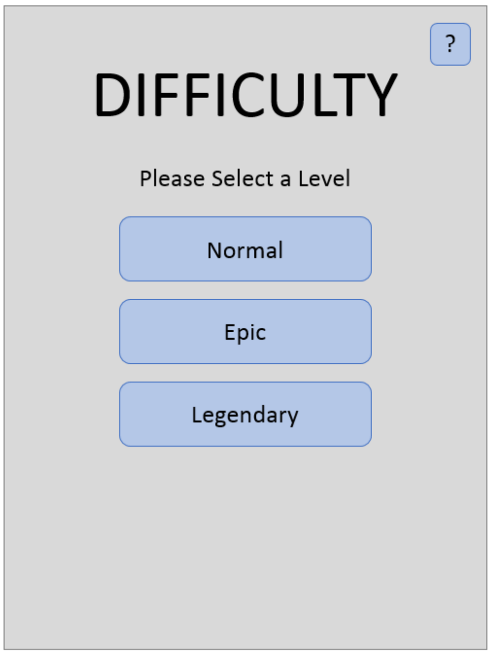
MAIN MENU

450x600



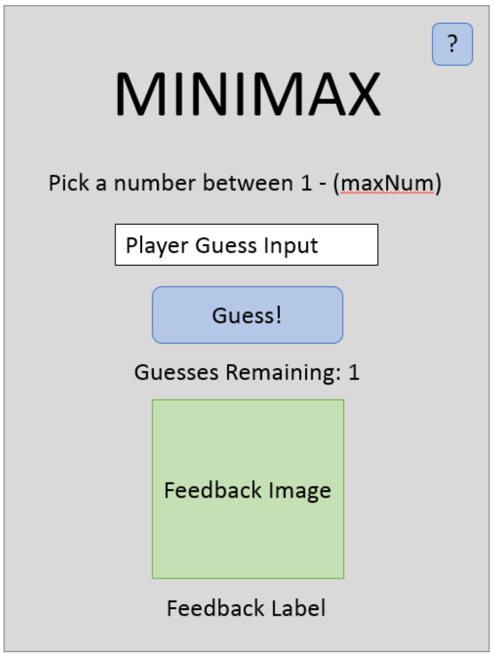
LEVEL SELECT

450x600



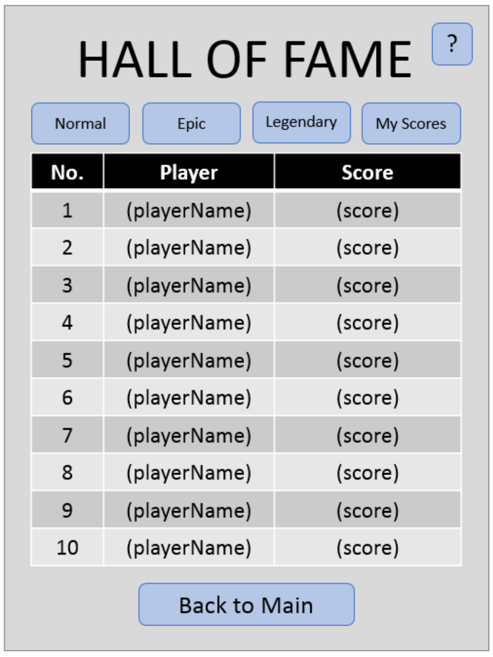
GAME

450x600



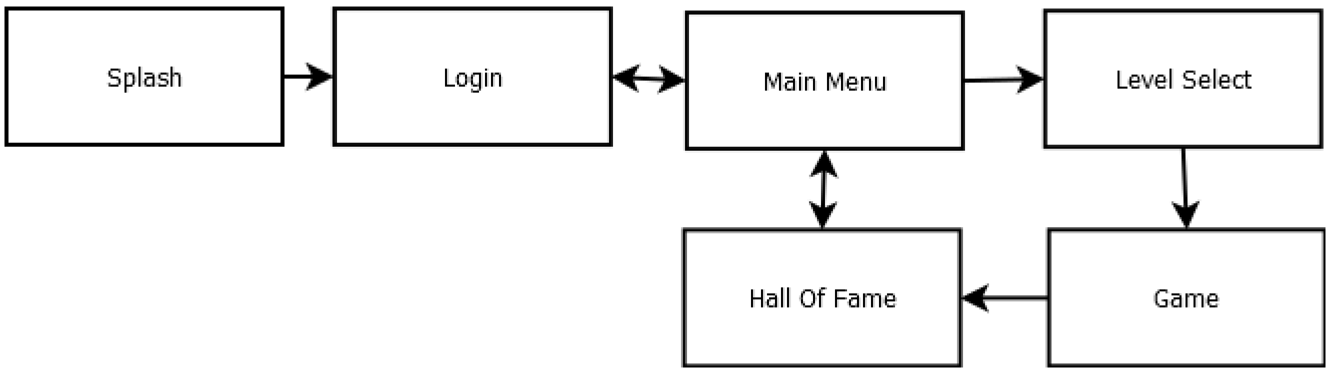
HALL OF FAME

450x600

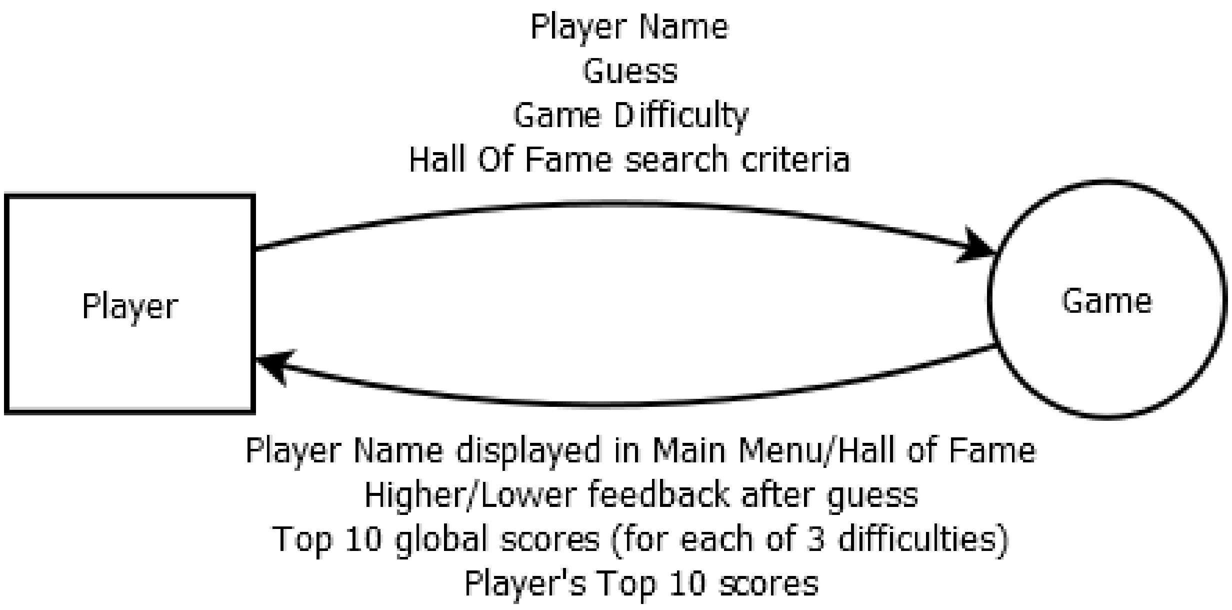


No.	Player	Score
1	(playerName)	(score)
2	(playerName)	(score)
3	(playerName)	(score)
4	(playerName)	(score)
5	(playerName)	(score)
6	(playerName)	(score)
7	(playerName)	(score)
8	(playerName)	(score)
9	(playerName)	(score)
10	(playerName)	(score)

STORYBOARD

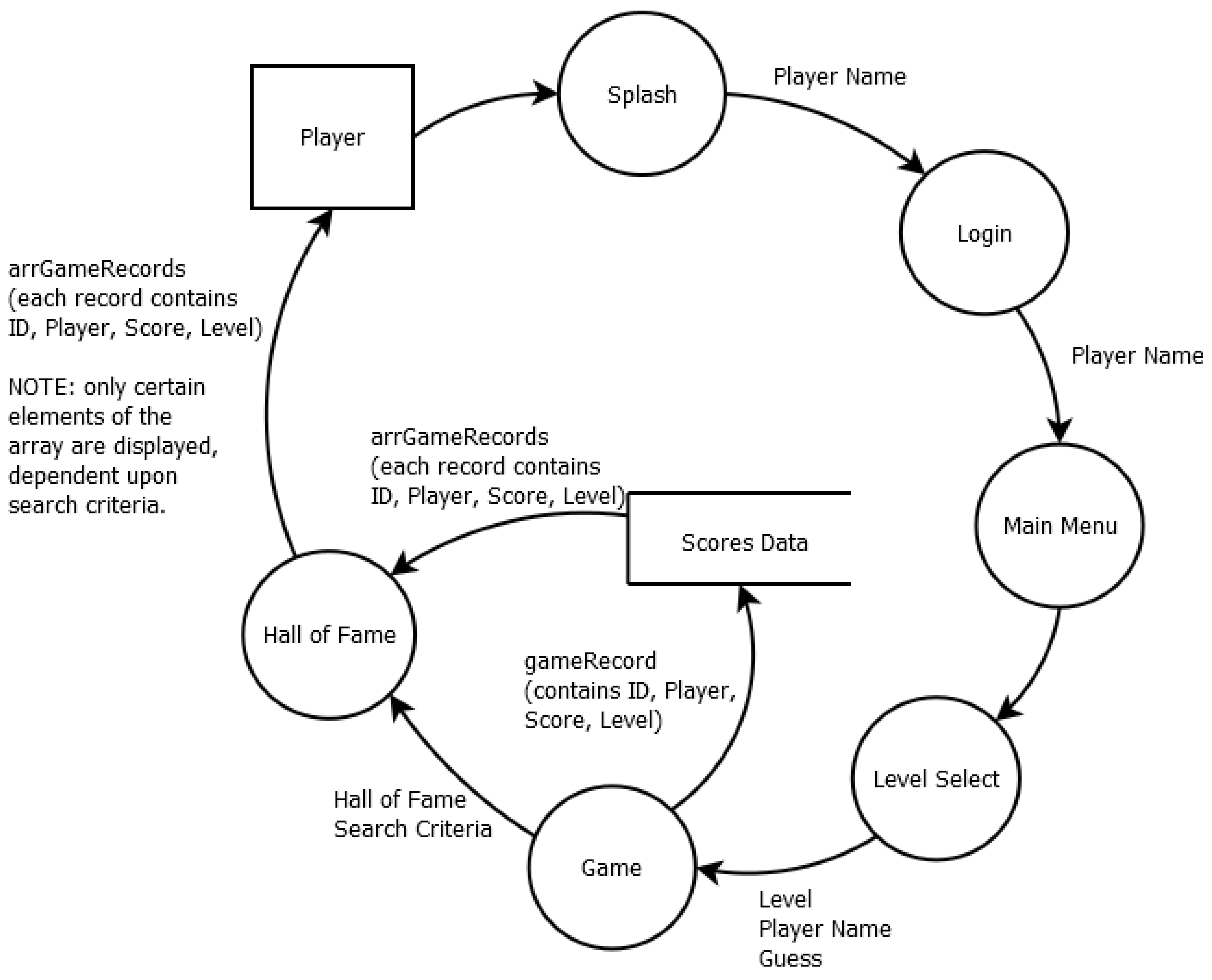


CONTEXT DIAGRAM

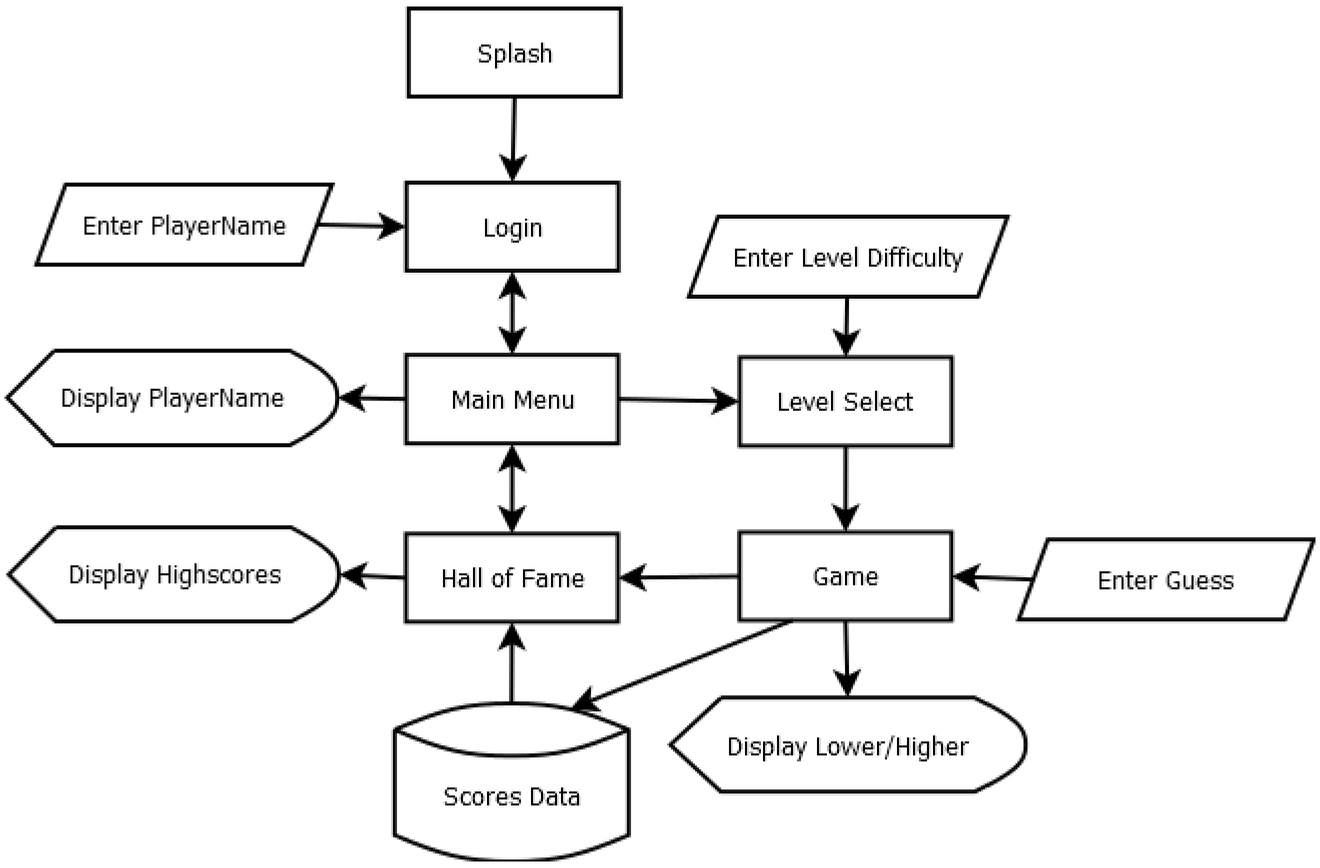

Player Name
Guess
Game Difficulty
Hall Of Fame search criteria
Player
Game
Player Name displayed in Main Menu/Hall of Fame
Higher/Lower feedback after guess
Top 10 global scores (for each of 3 difficulties)
Player's Top 10 scores

NOTE: Each "score" record comprises Game ID, Player Name, Game Difficulty, Scor

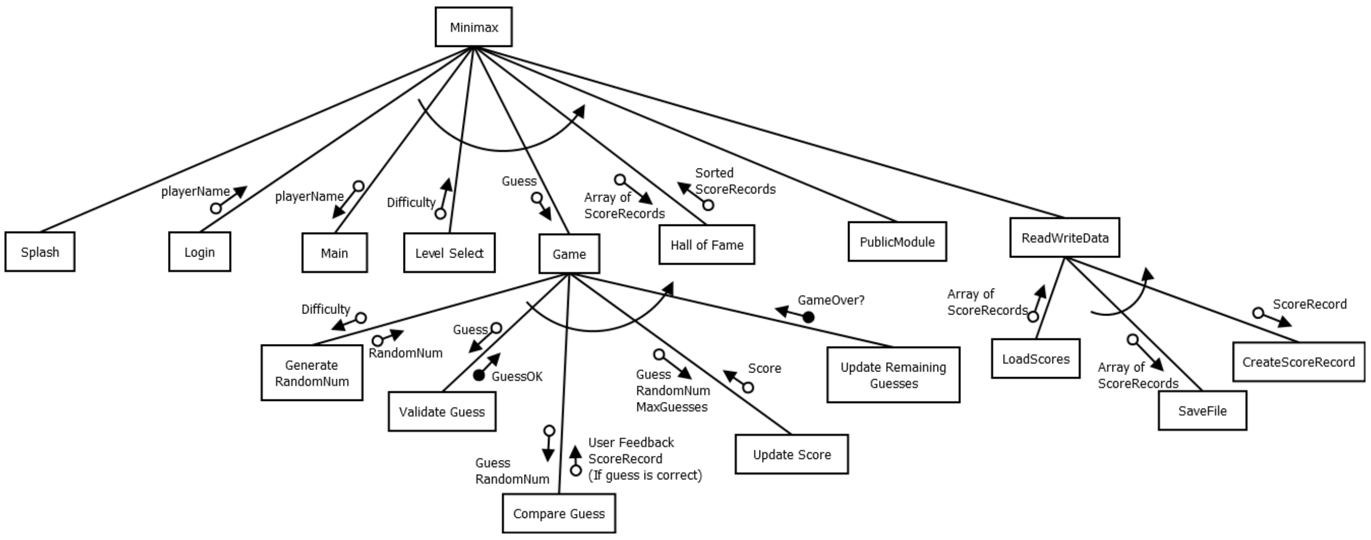
DATA FLOW DIAGRAM



SYSTEM FLOWCHART



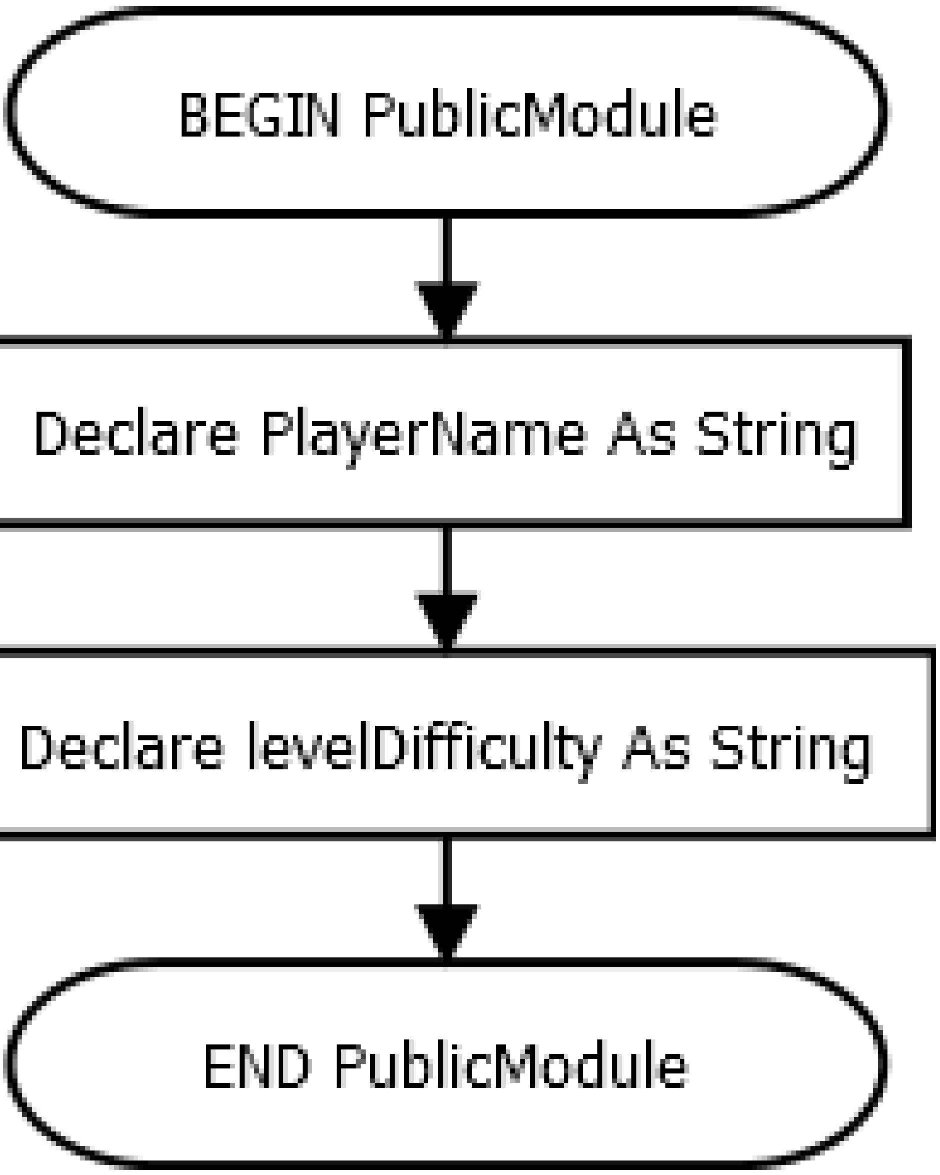
STRUCTURE CHART



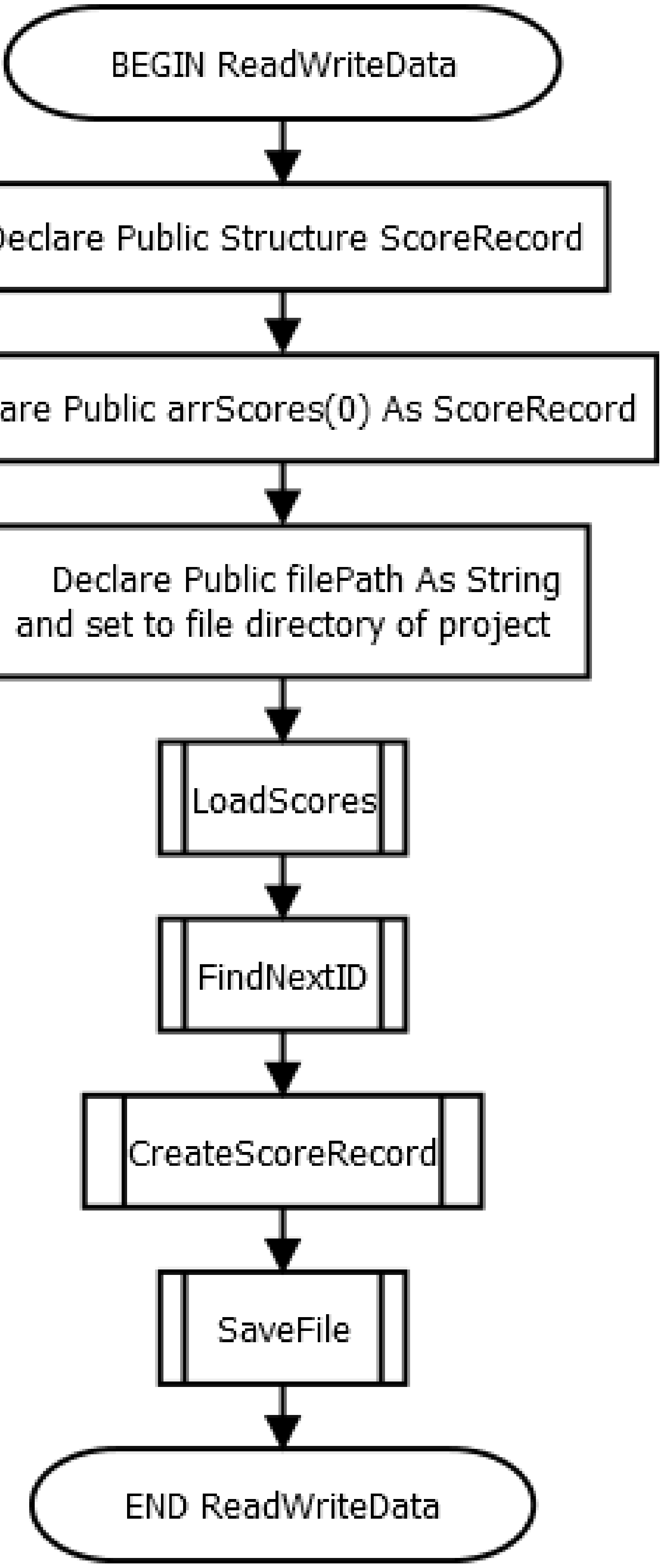
ALGORITHMS

FLOWCHARTS

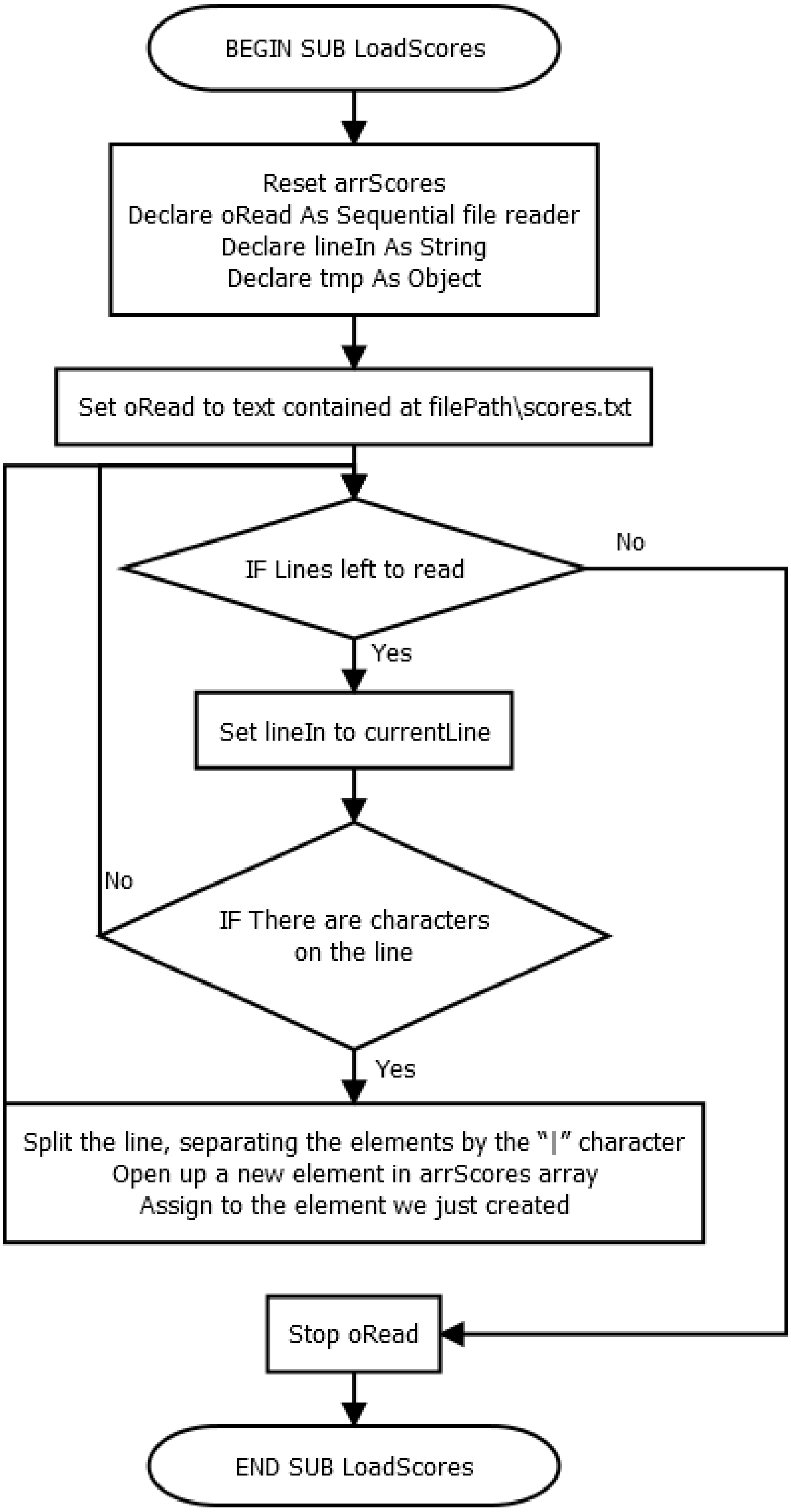
PublicModule

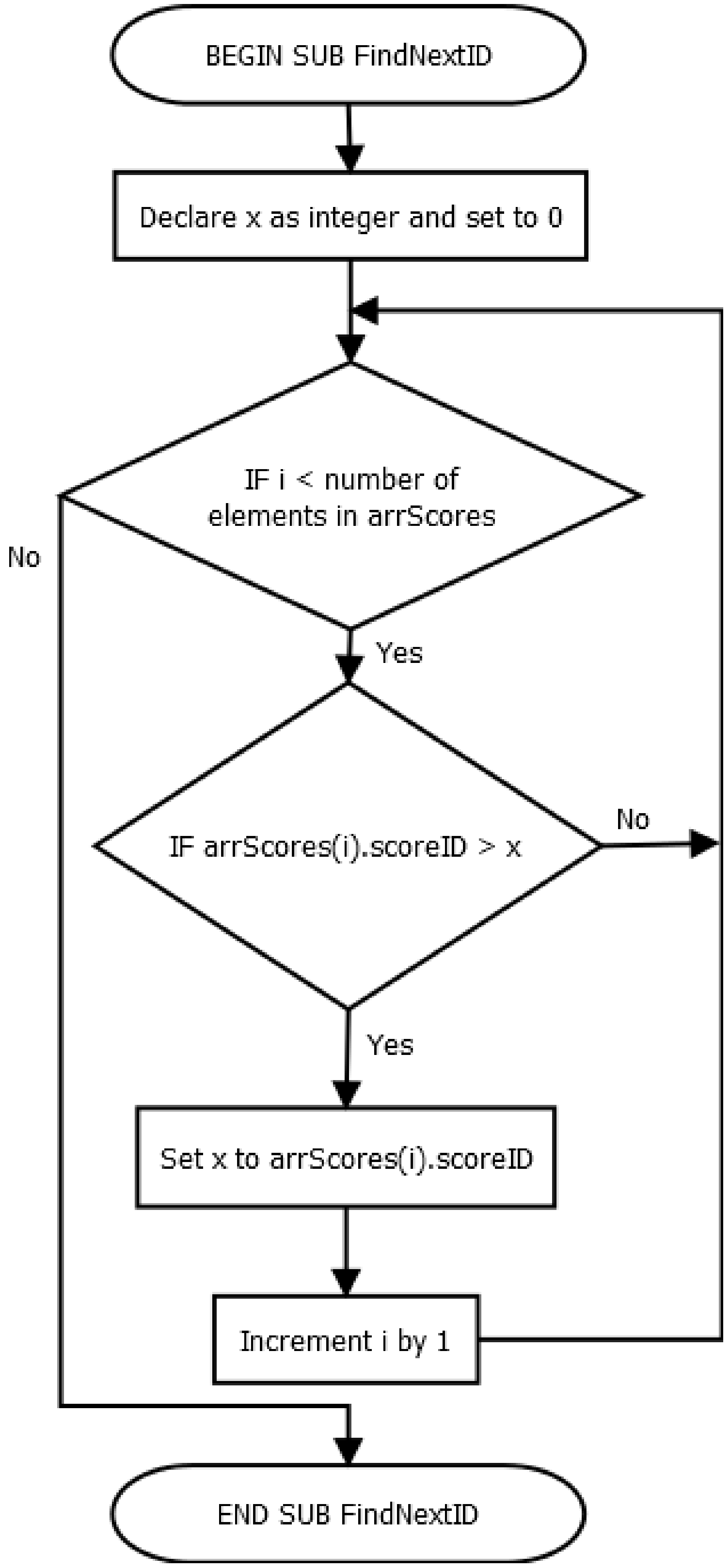

BEGIN PublicModule
Declare PlayerName As String
Declare levelDifficulty As String
END PublicModule

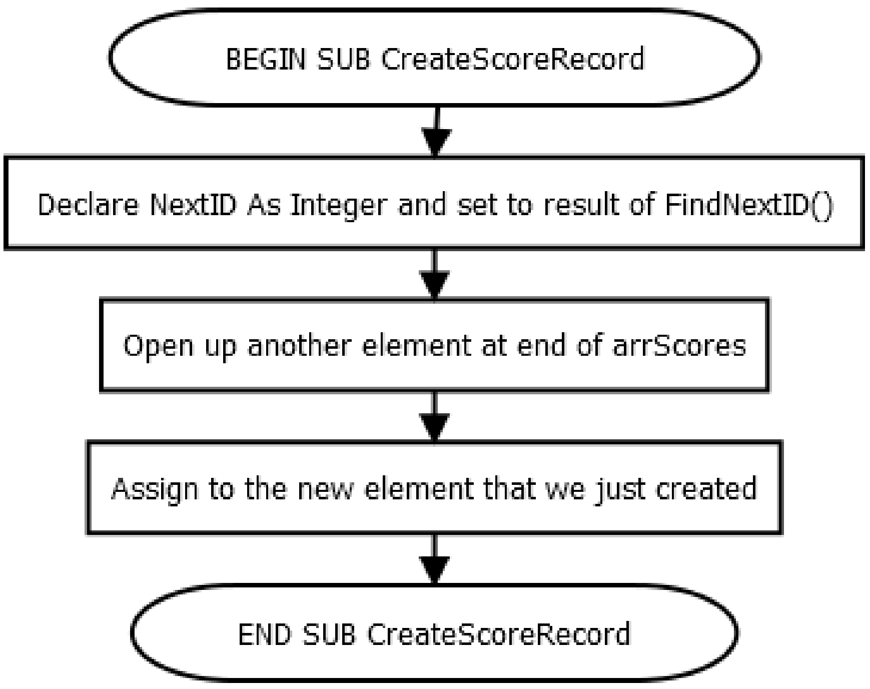
ReadWriteData

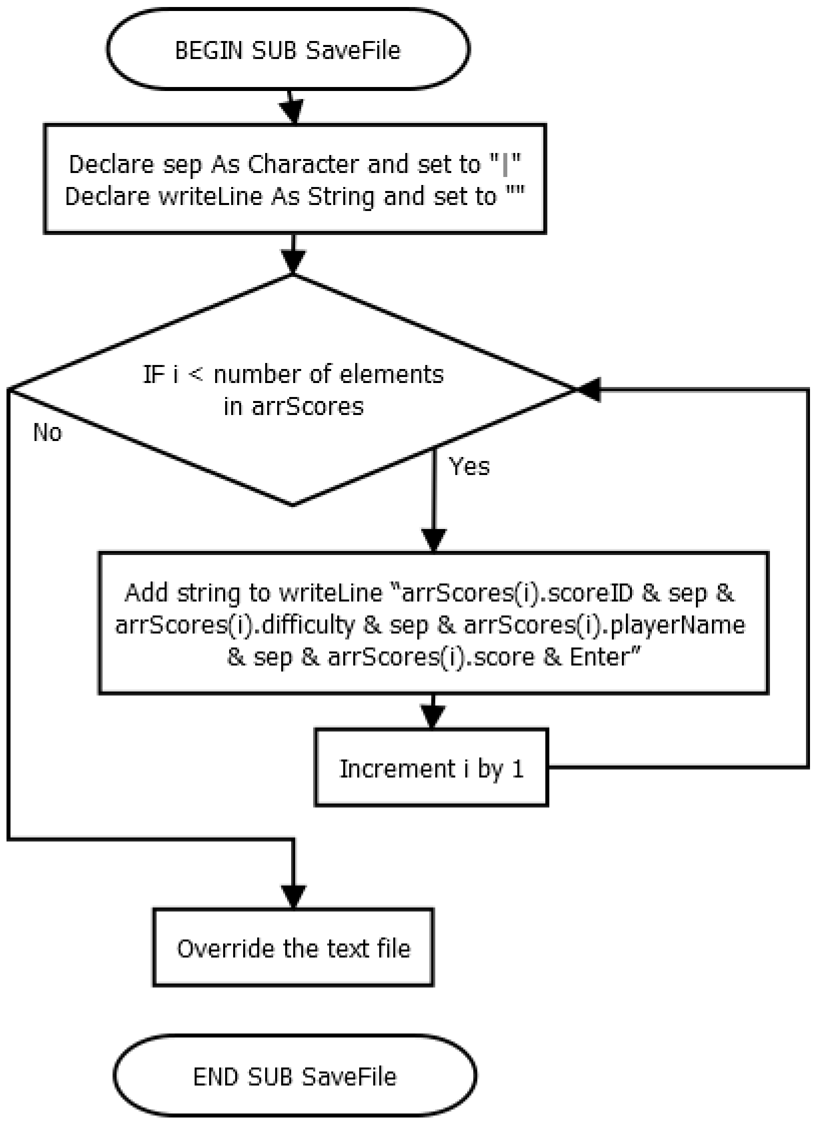


BEGIN ReadWriteData
Declare Public Structure ScoreRecord
Declare Public arrScores(0) As ScoreRecord
Declare Public filePath As String and set to file directory of project
LoadScores
FindNextID
CreateScoreRecord
SaveFile
END ReadWriteData

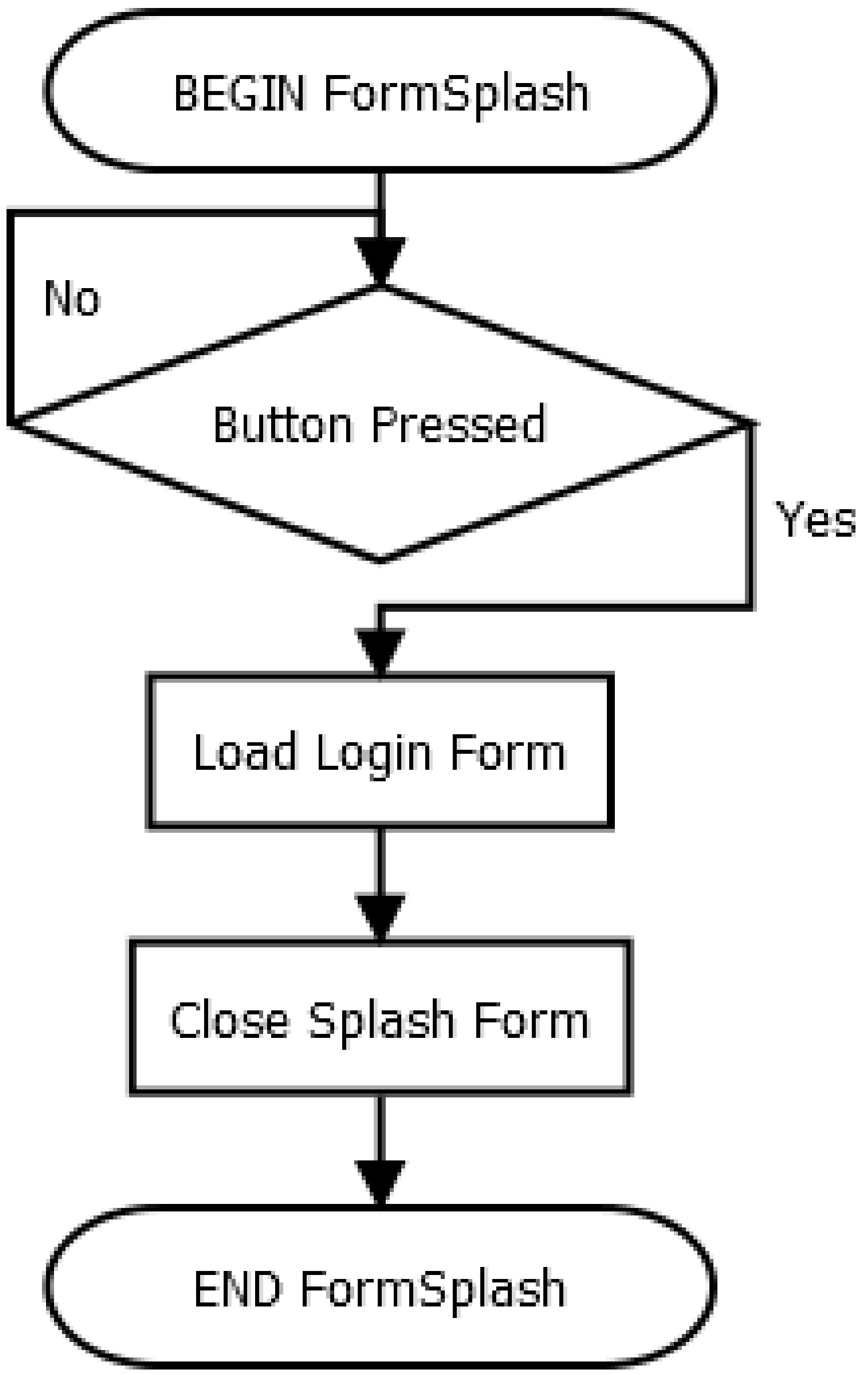

BEGIN SUB LoadScores
Reset arrScores
Declare oRead As Sequential file reader
Declare lineIn As String
Declare tmp As Object
Set oRead to text contained at filePath\scores.txt
IF Lines left to read
No
Yes
Set lineIn to currentLine
IF There are characters on the line
No
Yes
Split the line, separating the elements by the "|" character
Open up a new element in arrScores array
Assign to the element we just created
Stop oRead
END SUB LoadScores


BEGIN SUB FindNextID
Declare x as integer and set to 0
IF i < number of elements in arrScores
No
Yes
IF arrScores(i).scoreID > x
No
Yes
Set x to arrScores(i).scoreID
Increment i by 1
END SUB FindNextID

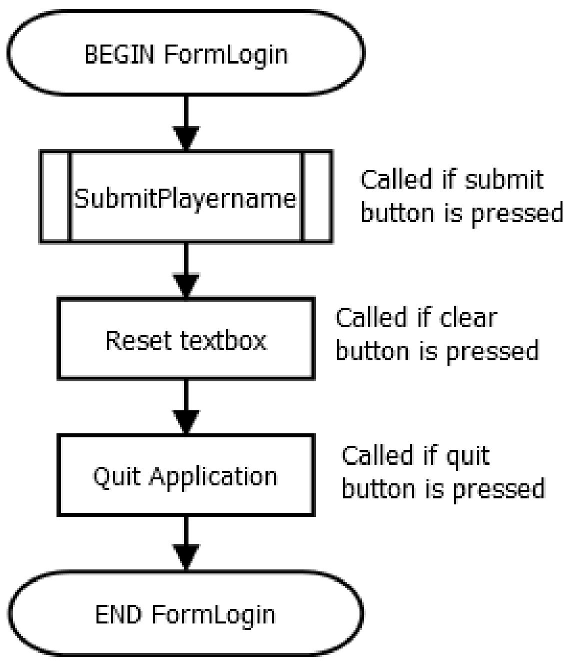

BEGIN SUB CreateScoreRecord
Declare NextID As Integer and set to result of FindNextID()
Open up another element at end of arrScores
Assign to the new element that we just created
END SUB CreateScoreRecord


BEGIN SUB SaveFile
Declare sep As Character and set to "|"
Declare writeLine As String and set to ""
IF i < number of elements in arrScores
No
Yes
Add string to writeLine "arrScores(i).scoreID & sep & arrScores(i).difficulty & sep & arrScores(i).playerName & sep & arrScores(i).score & Enter"
Increment i by 1
Override the text file
END SUB SaveFile

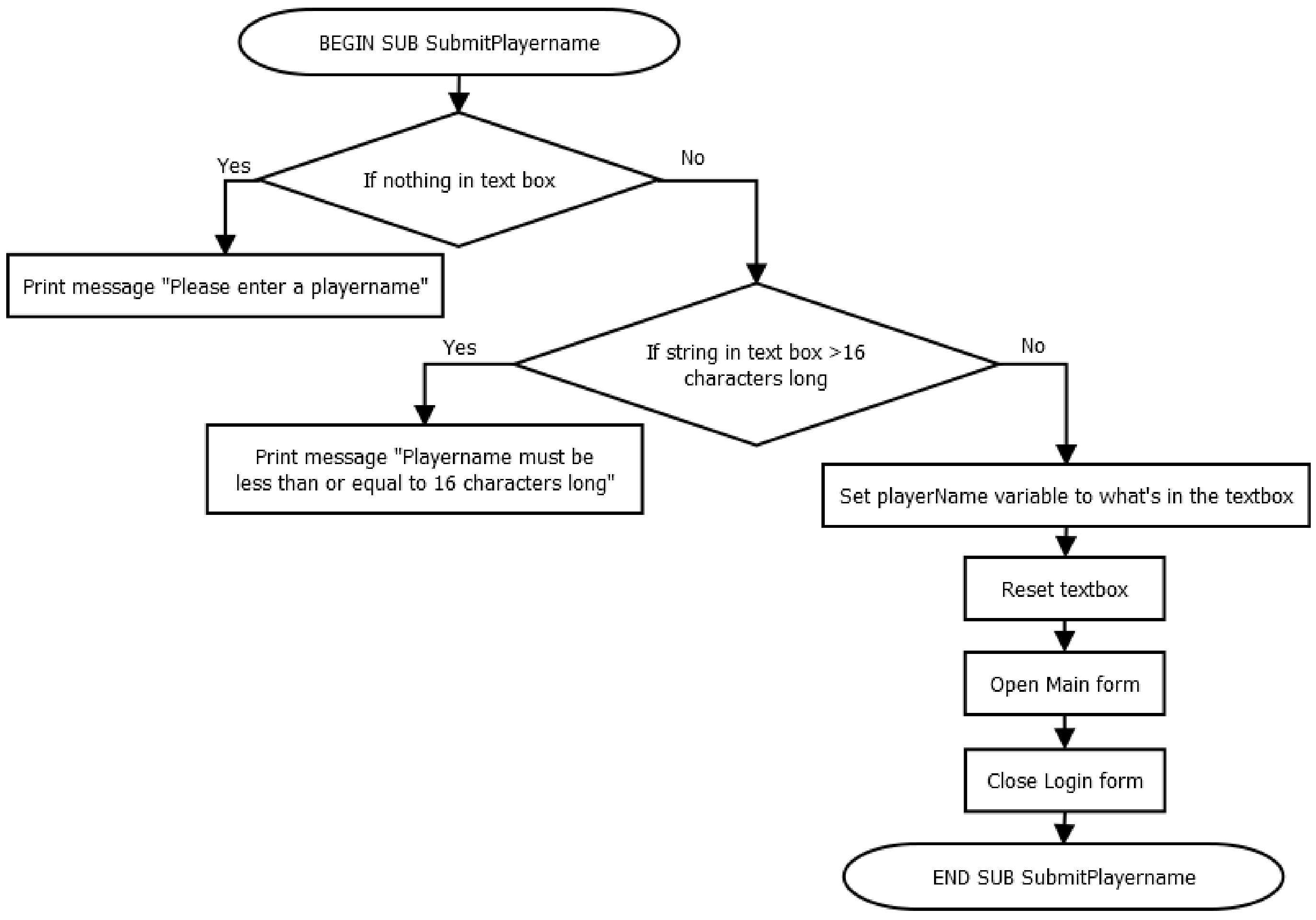
FormSplash


BEGIN FormSplash
No
Button Pressed
Yes
Load Login Form
Close Splash Form
END FormSplash

FormLogin

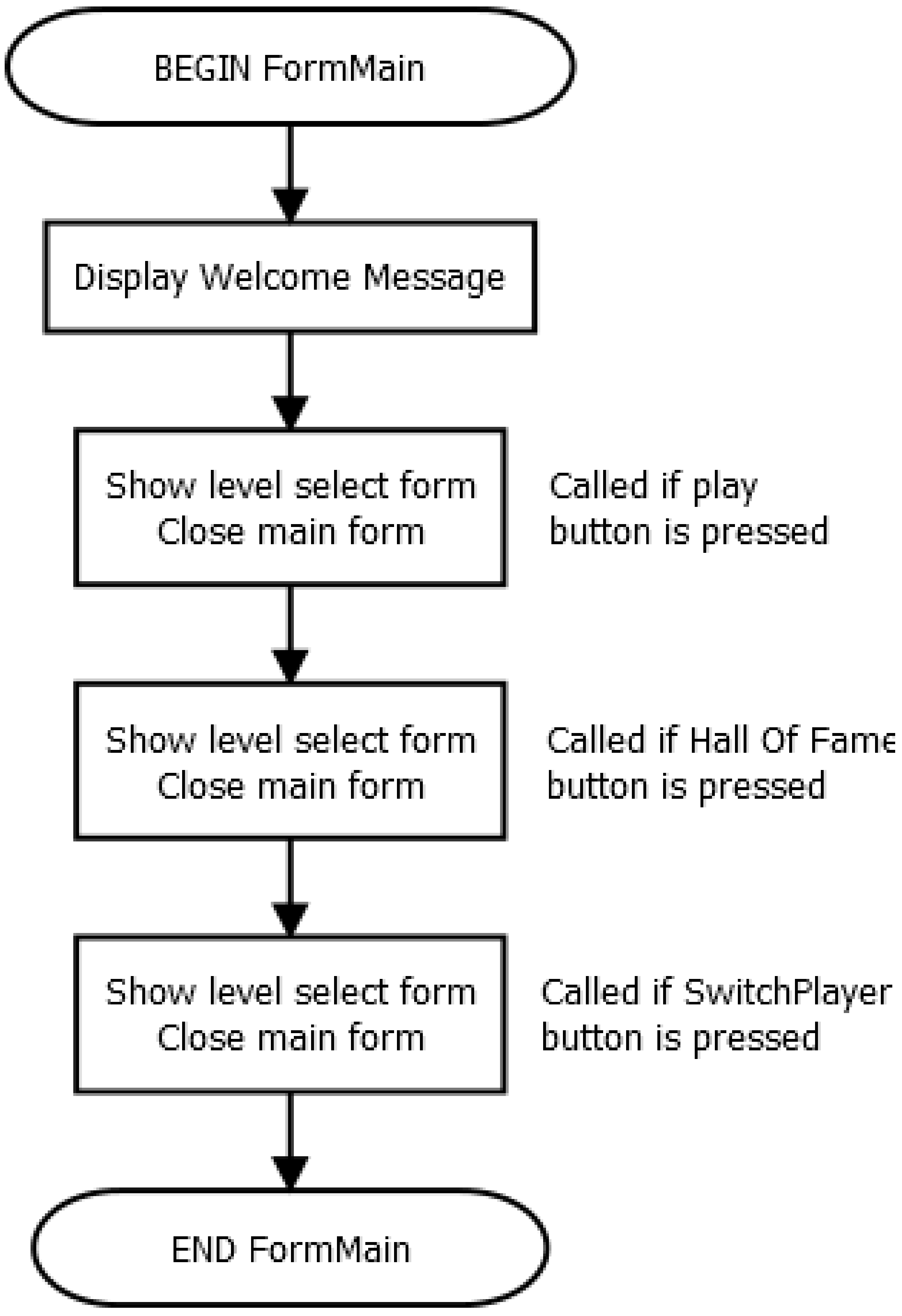


BEGIN FormLogin
SubmitPlayername
Called if submit button is pressed
Reset textbox
Called if clear button is pressed
Quit Application
Called if quit button is pressed
END FormLogin

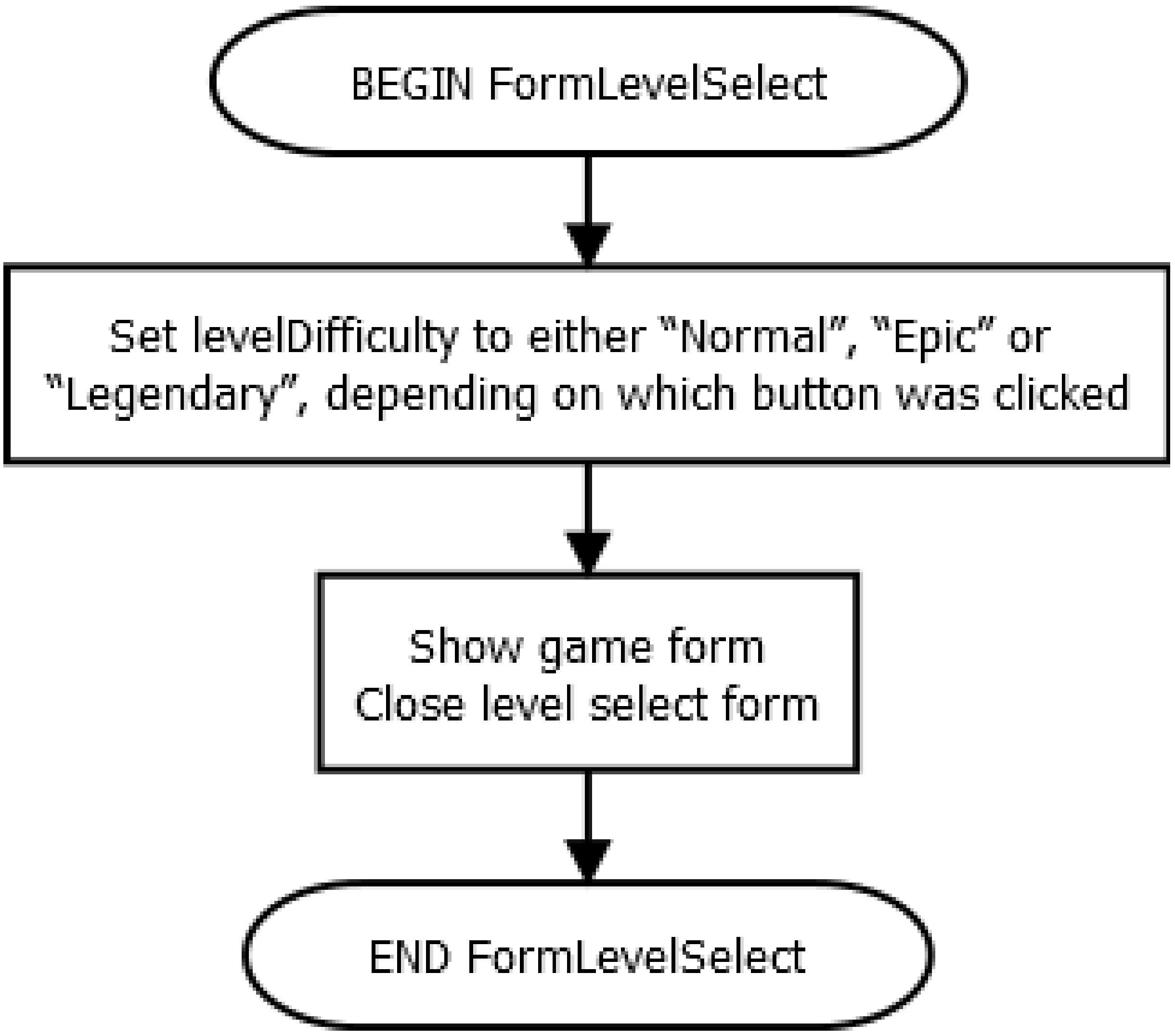


BEGIN SUB SubmitPlayername
If nothing in text box
Yes
No
Print message "Please enter a playername"
If string in text box >16 characters long
Yes
No
Print message "Playername must be less than or equal to 16 characters long"
Set playerName variable to what's in the textbox
Reset textbox
Open Main form
Close Login form
END SUB SubmitPlayername

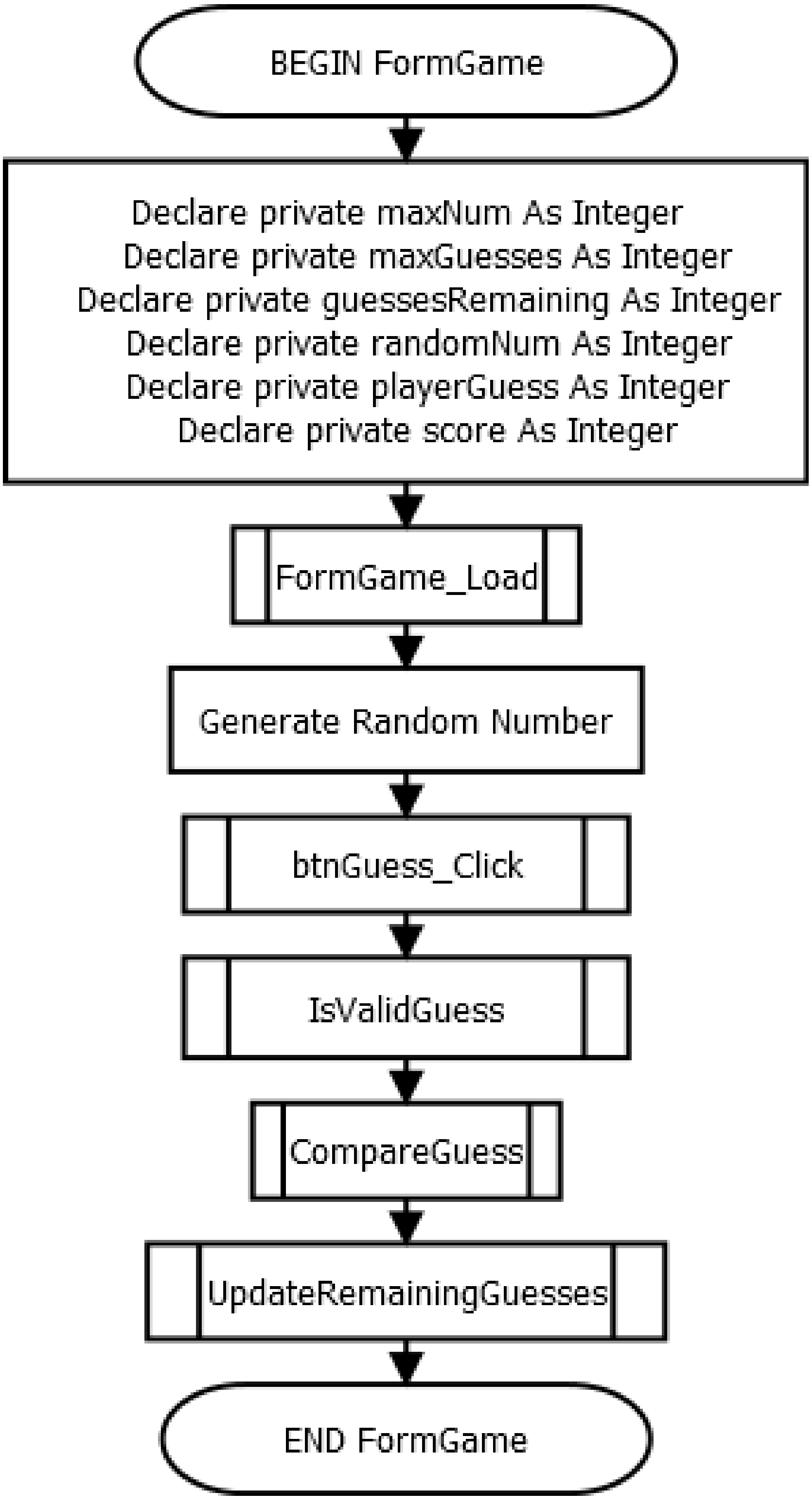
FormMain


BEGIN FormMain
Display Welcome Message
Show level select form
Close main form
Called if play
button is pressed
Show level select form
Close main form
Called if Hall Of Fame
button is pressed
Show level select form
Close main form
Called if SwitchPlayer
button is pressed
END FormMain

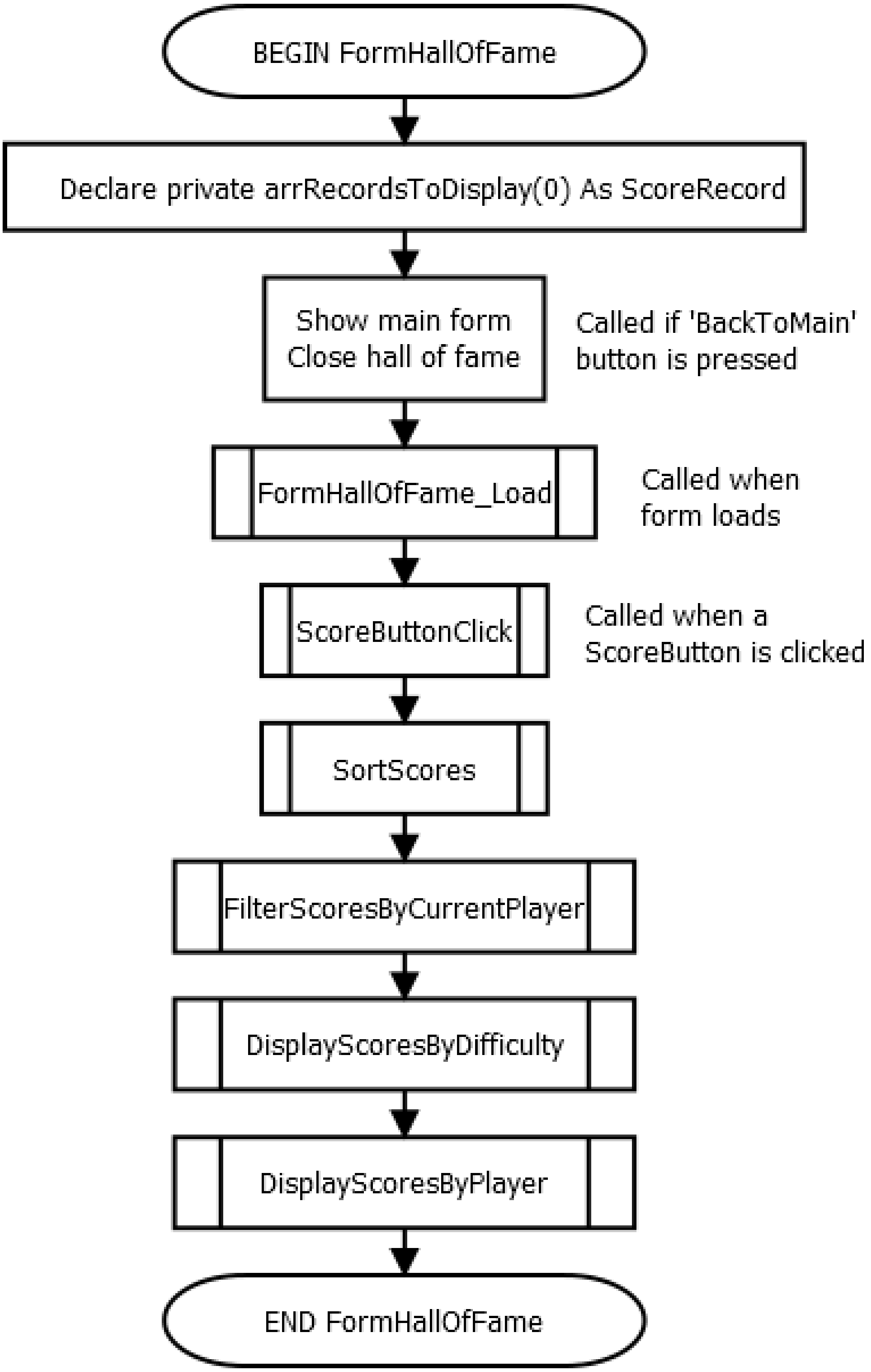
FormLevelSelect


BEGIN FormLevelSelect
Set levelDifficulty to either "Normal", "Epic" or "Legendary", depending on which button was clicked
Show game form
Close level select form
END FormLevelSelect

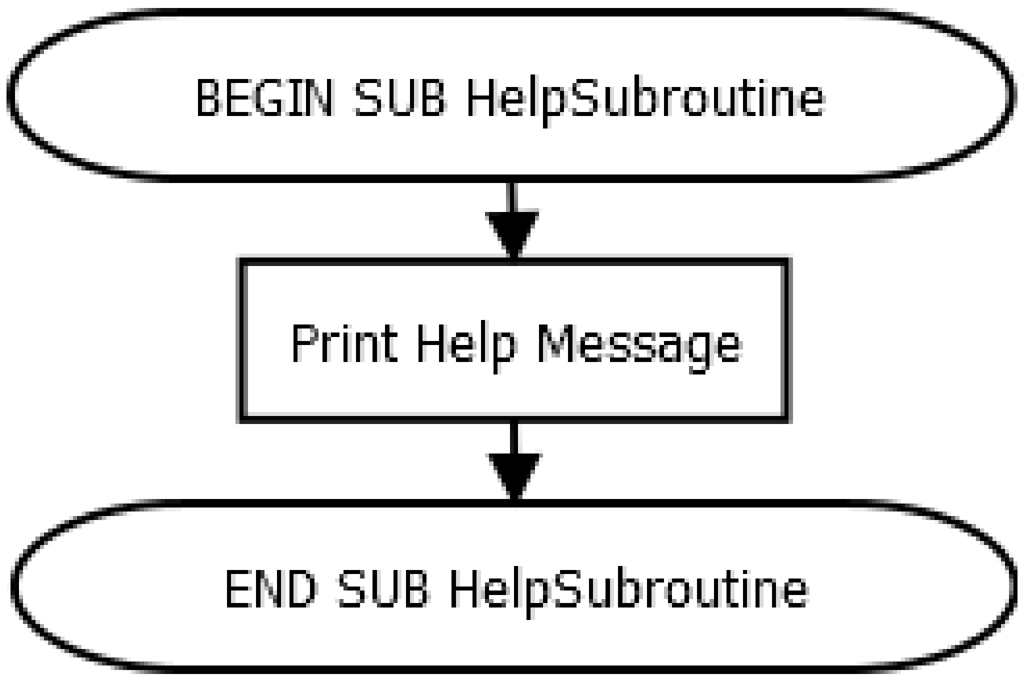
FormGame


BEGIN FormGame
Declare private maxNum As Integer
Declare private maxGuesses As Integer
Declare private guessesRemaining As Integer
Declare private randomNum As Integer
Declare private playerGuess As Integer
Declare private score As Integer
FormGame_Load
Generate Random Number
btnGuess_Click
IsValidGuess
CompareGuess
UpdateRemainingGuesses
END FormGame

FormHallOfFame


BEGIN FormHallOfFame
Declare private arrRecordsToDisplay(0) As ScoreRecord
Show main form
Close hall of fame
Called if 'BackToMain' button is pressed
FormHallOfFame_Load
Called when form loads
ScoreButtonClick
Called when a ScoreButton is clicked
SortScores
FilterScoresByCurrentPlayer
DisplayScoresByDifficulty
DisplayScoresByPlayer
END FormHallOfFame

HelpSubroutine



PSEUDOCODE

PublicModule

```
BEGIN MODULE PublicModule
   Declare public playerName As String
   Declare public levelDifficulty As String
END MODULE
```

ReadWriteData

```
BEGIN MODULE ReadWriteData
   Declare Public Structure ScoreRecord
      Declare scoreID As Integer
      Declare difficulty As String
      Declare playerName As String
      Declare score As Integer
   End Structure

   Declare Public arrScores(0) As ScoreRecord
   Declare Public filePath As String and set to file directory of project

   BEGIN SUB LoadScores()
      Reset arrScores
      Declare oRead As Sequential file reader
      Declare lineIn As String
      Declare tmp As Object

      Set oRead to text contained at filePath\scores.txt

      WHILE there are still lines left to read THEN
         Set lineIn to current line
         IF There are characters on the line THEN
            Split the line, separating the elements by the "|" character
            Open up a new element in arrScores array
            Assign to the scoreID part of the element in arrScores that we just created
            Assign to the difficulty part of the element in arrScores that we just created
            Assign to the playerName part of the element in arrScores that we just created
            Assign to the score part of the element in arrScores that we just created
         END IF
      END WHILE
      Stop oRead
   END SUB

   BEGIN FUNCTION FindNextID()
      Declare x As Integer and set to 0
      FOR i = 0 TO Number of elements in arrScores
         IF arrScores(i).scoreID > x THEN
            Set x to arrScores(i).scoreID
         END IF
      Next
      Set return value FindNextID to x + 1
   END FUNCTION

   BEGIN SUB CreateScoreRecord(difficulty As String, playerName As String, score As Integer)
      Declare NextID As Integer and set to result of FindNextID()
      Open up another element at end of arrScores
      Assign to the scoreID field of the new element in the array that we just created.
      Assign to the difficulty field of the new element in the array that we just created.
      Assign to the playerName field of the new element in the array that we just created.
      Assign to the score field of the new element in the array that we just created.
   END SUB

   PUBLIC SUB SaveFile()
      Declare sep As Character and set to "|"
      Declare writeLine As String and set to ""
      FOR i = 0 TO Number of elements in arrScores
         Add string to writeLine “arrScores(i).scoreID & sep & arrScores(i).difficulty & sep &
arrScores(i).playerName & sep & arrScores(i).score & Enter”
      NEXT
      Override the text file
   END SUB
END MODULE
```

FormSplash

```
BEGIN CLASS formSplash
   BEGIN SUB StartGame(sender As Object, e As Event) When start button is clicked.
      Show login form
      Hide splash form
   END SUB
END CLASS
```

FormLogin

```
BEGIN CLASS formLogin
   BEGIN SUB SubmitPlayername(sender As Object, e As Event) When submit button is clicked.
      IF Nothing in the text box THEN
         Print message "Please Enter a Playername!"
      ELSE IF string in text box is > 16 characters long THEN
         Print Message "Playername must be less than or equal to 16 characters long"
      ELSE
         Set playerName variable to what's in the textbox
         Reset textbox
         Show main form
         Close splash form
      END IF
   END SUB

   BEGIN SUB ClearPlayername(sender As Object, e As Event) When clear button is clicked
      Reset textbox
   END SUB

   BEGIN SUB btnQuit_Click(sender As Object, e As Event) When quit button is clicked
      End Program
   END SUB
END CLASS
```

FormMain

```
BEGIN CLASS formMain
   BEGIN SUB FormMainLoadEvent(sender As Object, e As Event) When main form loads
      Display welcome message
   END SUB

   BEGIN SUB btnPlay_Click(sender As Object, e As Event) When play button is clicked
      Show level select form
      Close main form
```

```
  END SUB

  BEGIN SUB btnHallOfFame_Click(sender As Object, e As Event) When Hall of fame button is clicked
    Show hall of fame form
    Close main form
  END SUB

  BEGIN SUB btnSwitchPlayer_Click(sender As Object, e As Event) When Switch Player button is
clicked
    Show login form
    Close main form
  END SUB
END CLASS
```

FormLevelSelect

```
BEGIN CLASS formLevelSelect
  BEGIN SUB LevelSelect(sender As Object, e As Event) When "Normal", "Epic" or "Legendary" button
is clicked
    Set levelDifficulty to either "Normal", "Epic" or "Legendary", depending on which button was clicked
    Show game form
    Close level select form
  END SUB
END CLASS
```

FormGame

```
BEGIN CLASS formGame
  Declare private maxNum As Integer
  Declare private maxGuesses As Integer
  Declare private guessesRemaining As Integer
  Declare private randomNum As Integer
  Declare private playerGuess As Integer
  Declare private score As Integer

  BEGIN SUB formGame_Load(sender As Object, e As Event) When game form loads
    IF levelDifficulty is Normal THEN
      Set maxNum to 10
      Set maxGuesses to 3
    ELSE IF levelDifficulty is Epic THEN
      Set maxNum to 100
      Set maxGuesses to 5
    ELSE IF levelDifficulty is Legendary THEN
      Set maxNum to 1000
      Set maxGuesses to 9
    END IF
    Set guessesRemaining to maxGuesses
    Update GuessesRemaining label
    Update GameInstructions label
    Clear input box
    Generate a new random number by calling GenerateRandomNum, passing in 1 and maxNum
    Set score to maxNum
    Load scores into RAM by calling LoadScores()
  END SUB

  BEGIN FUNCTION GenerateRandomNum(lowerBound As Integer, upperBound As Integer)
    Update random float in system
    Declare temp As Integer
    Set temp to a random number between lowerBound and upperBound
    Return temp
 END FUNCTION

 BEGIN SUB btnGuess_Click(sender As Object, e As Event) When guess button is clicked
    Declare guessOK As Boolean
    Set guessOK to result of IsValidGuess()
    IF guessOK is True THEN
       Update score by calling UpdateScore()
       Compare guess by calling CompareGuess()
    END IF
 END SUB

 BEGIN FUNCTION IsValidGuess()
    TRY
       Set playerGuess to what's in the text box
       IF playerGuess < 1 OR playerGuess > maxNum THEN
          Print message "Please insert a number between 1 and maxNum"
          Reset text box
          Return False
       ELSE
          Return True
       END IF
    CATCH ex As Exception
       Print message "Please insert a number between 1 and maxNum"
       Reset text box
       Return False
    END TRY
 END FUNCTION

 BEGIN SUB CompareGuess()
    IF playerGuess > randomNum THEN
       Update feedback label to "Lower!"
       Set visibility of down arrow to true
       Set visibility of up arrow to false
       Update remaining guesses by calling UpdateRemainingGuesses()
    ELSEIF playerGuess < randomNum THEN
       Update feedback label to "Higher!"
       Set visibility of down arrow to false
       Set visibility of up arrow to true
       Update remaining guesses by calling UpdateRemainingGuesses()
    ELSE
       Update feedback label to "You Win!"
       Set visibility of up arrow to false
       Set visibility of down arrow to false
       Add new scoreRecord to arrScores, done by calling CreateScoreRecord(levelDifficulty,
playerName, score)
       Save arrScores to file by calling SaveFile()
       Print message "You Win! Your score was: score! Click OK to proceed to Hall of Fame."
       Show hall of fame form
       Close game form
    END IF
 END SUB

 BEGIN SUB UpdateScore()
    Set score to score - RoundUp(AbsoluteValue((playerGuess - randomNum) / (maxGuesses - 1)))
 END SUB

 BEGIN SUB UpdateRemainingGuesses()
```

```
        IF guessesRemaining > 0 THEN
            Subtract 1 from remaining guesses
            Update guessesRemaining label
            IF guessesRemaining = 0 THEN
                Update feedback label to "Out of Guesses!"
                Set visibility of up arrow to false
                Set visibility of down arrow to false
                Print message "Out of Guesses! The number was randomNum! Click OK to return to the Main
Menu.")
                Show main form
                Close game form
            END IF
        END IF
    END SUB
END CLASS
```

FormHallOfFame

```
BEGIN CLASS formHallOfFame
    Declare private arrRecordsToDisplay(0) As ScoreRecord
    BEGIN SUB btnBackToMain_Click(sender As Object, e As Event) When BackToMain button is
clicked
        Show main form
        Close hall of fame
    END SUB

    BEGIN SUB formHallOfFame_Load(sender As Object, e As Event) When Hall of fame loads
        Load scores from file into RAM by calling LoadScores()
        Set arrRecordsToDisplay to arrScores
        Set arrRecordsToDisplay to result of SortScores(arrRecordsToDisplay)
        Set arrRecordsToDisplay to result of FilterScoresByCurrentPlayer(arrRecordsToDisplay)
        Display scores in list box by calling DisplayScoresByPlayer(arrRecordsToDisplay)
    END SUB

    BEGIN SUB ScoreButtonClick(sender As Object, e As Event) When either “Normal”, “Epic”,
“Legendary” or “MyScores” buttons are clicked
        Load scores from file into RAM by calling LoadScores()
        Set arrRecordsToDisplay to arrScores
        Set arrRecordsToDisplay to result of SortScores(arrRecordsToDisplay)
        IF Clicked button’s tag is 3 THEN
            Set arrRecordsToDisplay to result of FilterScoresByCurrentPlayer(arrRecordsToDisplay)
            Display scores in list box by calling DisplayScoresByPlayer(arrRecordsToDisplay)
        ELSE
            Set arrRecordsToDisplay to result of FilterScoresByDifficulty(arrRecordsToDisplay, Text of
button)
            Display scores in list box by calling DisplayScoresByDifficulty(arrRecordsToDisplay, Text of
button)
        END IF
    END SUB

    BEGIN FUNCTION SortScores(arrItems() As ScoreRecord)
        Declare firstIndex As Integer and set to 0
        Declare lastIndex As Integer and set to number of elements in arrItems
        Declare endUnsortedIndex As Integer and set to firstIndex
        Declare currentIndex As Integer
        Declare maxItem As ScoreRecord
        Declare maxIndex As Integer
        Declare temp As ScoreRecord
        WHILE endUnsortedIndex < lastIndex THEN
            Set currentIndex to endUnsortedIndex
            Set maxItem to arrItems(currentIndex)
            Set maxIndex to currentIndex
            WHILE currentIndex < lastIndex THEN
              Increment currentIndex by 1
              IF arrItems(currentIndex).ScoreValue > maxItem.ScoreValue THEN
                Set maxItem to arrItems(currentIndex)
                Set maxIndex to currentIndex
              END IF
            END WHILE
            Set temp to arrItems(maxIndex)
            Set arrItems(maxIndex) to arrItems(endUnsortedIndex)
            Set arrItems(endUnsortedIndex) to temp
            Increment endUnsortedIndex by 1
        END WHILE
        Return arrItems
    END FUNCTION

    BEGIN FUNCTION FilterScoresByCurrentPlayer(arrScoresToFilter() As ScoreRecord)
        Declare arrToReturn(0) As ScoreRecord
        FOR i = 0 TO Number of elements in arrScoresToFilter
            IF arrScoresToFilter(i).playerName is the same as current playerName THEN
              Declare newID As Integer and set to Number of elements in arrToReturn + 1
              Create a new element at arrToReturn(newID)
              Set arrToReturn(newID) to arrScoresToFilter(i)
            END IF
        NEXT
        Return arrToReturn
    END FUNCTION

    BEGIN FUNCTION FilterScoresByDifficulty(arrScoresToFilter() As ScoreRecord, filterDifficulty As
String)
        Declare arrToReturn(0) As ScoreRecord
        FOR i = 0 TO Number of elements in arrScoresToFilter
            IF arrScoresToFilter(i).difficulty is equal to filterDifficulty THEN
              Declare newID As Integer and set to Number of elements in arrToReturn + 1
              Create a new element at arrToReturn(newID)
              Set arrToReturn(newID) to arrScoresToFilter(i)
            END IF
        NEXT
        Return arrToReturn
    END FUNCTION

    BEGIN SUB DisplayScoresByDifficulty(arrToDisplay() As ScoreRecord, difficulty As String)
        Clear list box
        Display the difficulty that has been selected in the listbox
        Display empty line in list box
        Display in list box “NO: & Tab & PLAYER: & Tab & Tab & SCORE:"
        IF Number of elements in arrToDisplay <= 10 THEN
            FOR i = 1 TO Number of elements in arrToDisplay
              IF Length of arrToDisplay(i).playerName < 9 THEN
                Additem to list box “i & Tab & arrToDisplay(i).playerName & Tab & Tab &
arrToDisplay(i).score)
              ELSE
                Add item to list box “i & Tab & arrToDisplay(i).playerName & Tab & arrToDisplay(i).score”
              END IF
            NEXT
        ELSE
            FOR i = 1 TO 10
              IF Length of arrToDisplay(i).playerName < 9 THEN
                Add item to list box “i & Tab & arrToDisplay(i).playerName & Tab & Tab &
arrToDisplay(i).score)
              ELSE
                Add item to list box “i & Tab & arrToDisplay(i).playerName & Tab & arrToDisplay(i).score”
              END IF
            NEXT
        END IF
    END SUB

    BEGIN SUB DisplayScoresByPlayer(arrToDisplay() As ScoreRecord)
        Clear list box
        Display item in list box “My Scores"
        Display empty line in list box
        IF number of elements in arrToDisplay is equal to 0 THEN
            Display item in list box "You have not set any highscores yet!"
        ELSE
            Display item in list box "NO: & Tab & DIFFICULTY: & Tab & SCORE:"
            FOR i = 1 TO Number of elements in arrToDisplay
              IF Length of arrToDisplay(i).playerName < 9 THEN
                Add item to list box “i & Tab & arrToDisplay(i).difficulty & Tab & Tab & arrToDisplay(i).score)
              ELSE
                Add item to list box “i & Tab & arrToDisplay(i).difficulty & Tab & arrToDisplay(i).score)
              END IF
            NEXT
        END IF
    END SUB
END CLASS
```

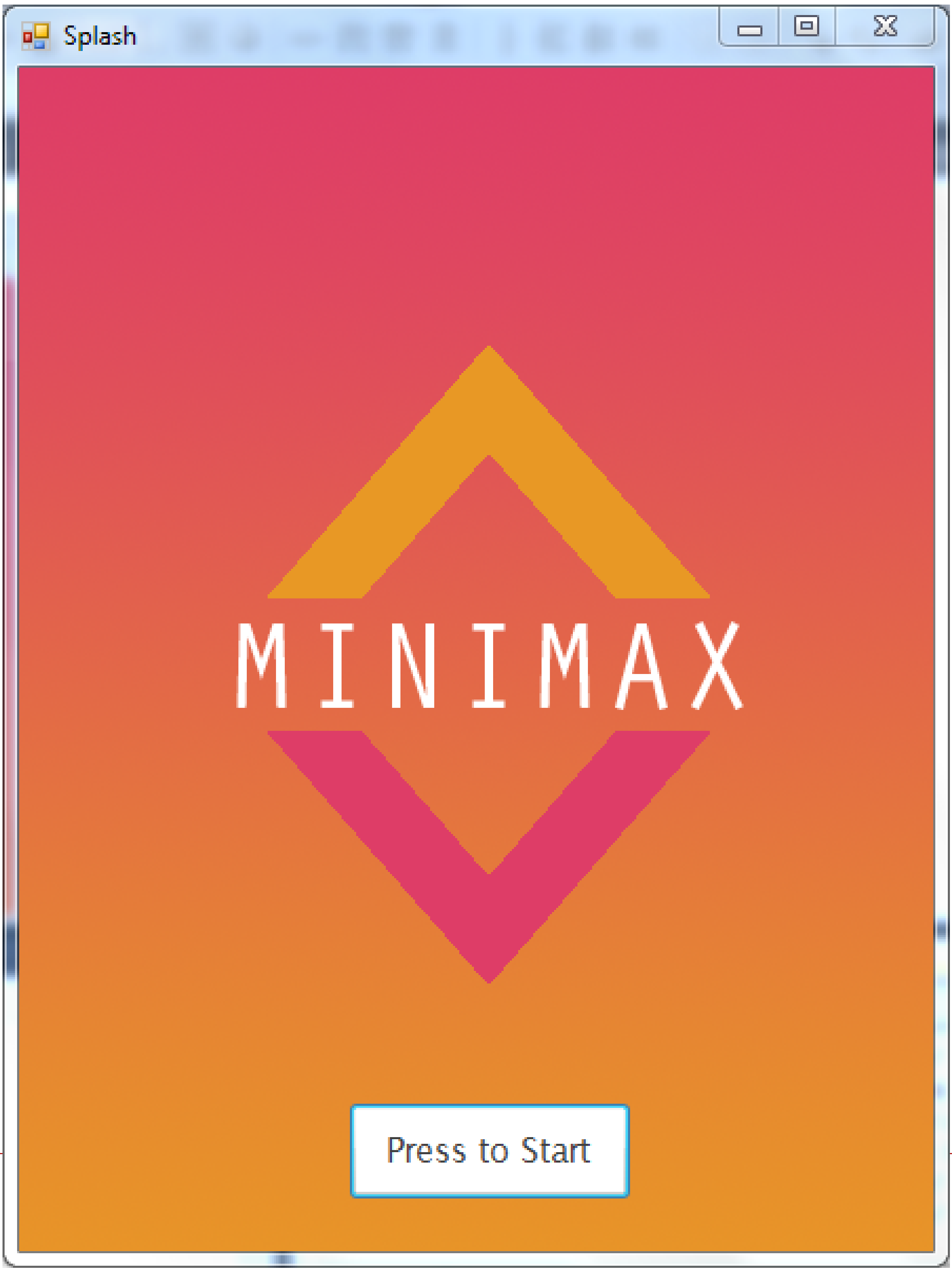
HelpSubroutine

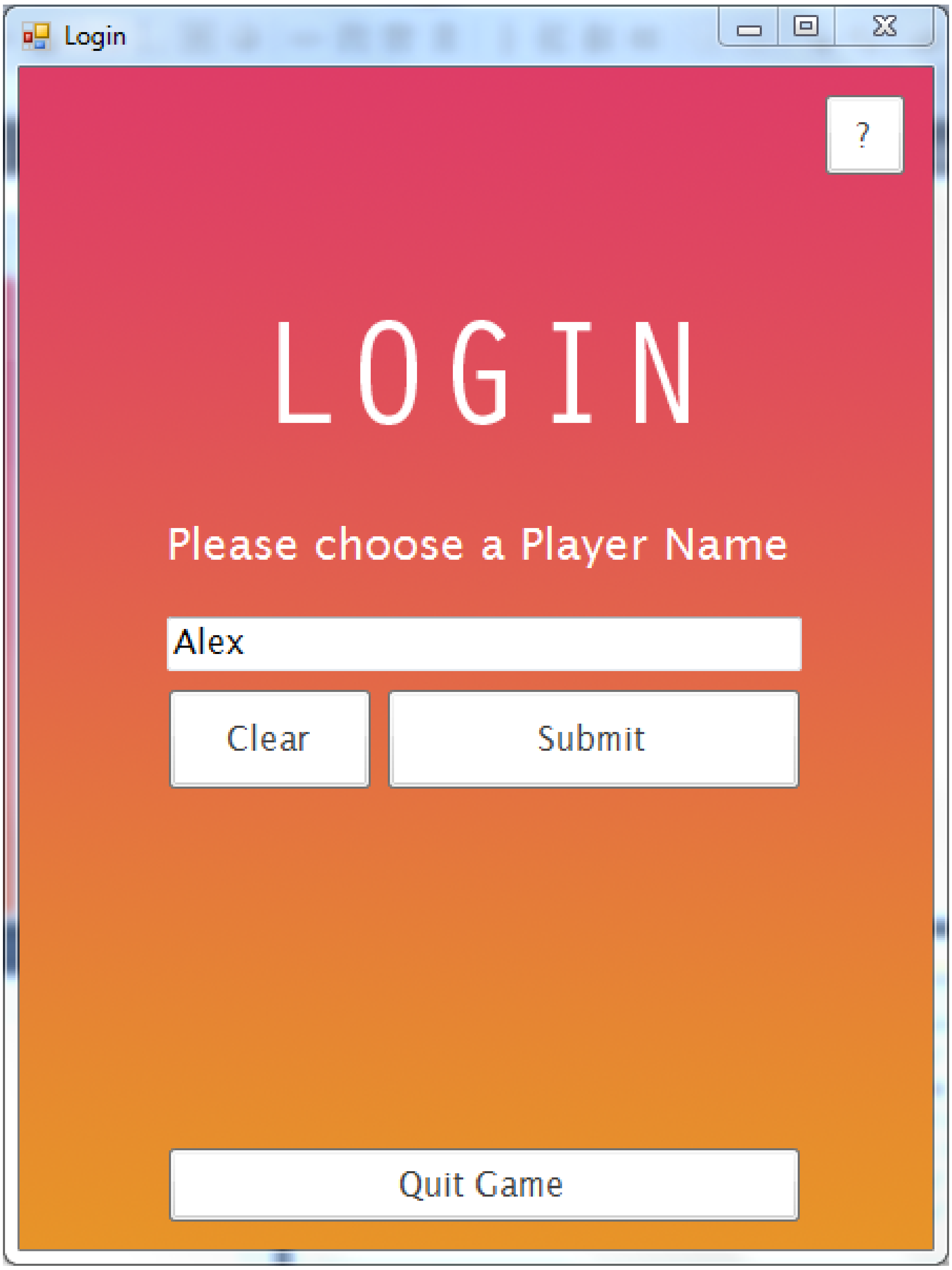
This subroutine is a general subroutine implemented in most forms, and forms the basis for the online help system. The help message that is displayed is different depending on which form the help button is located on.

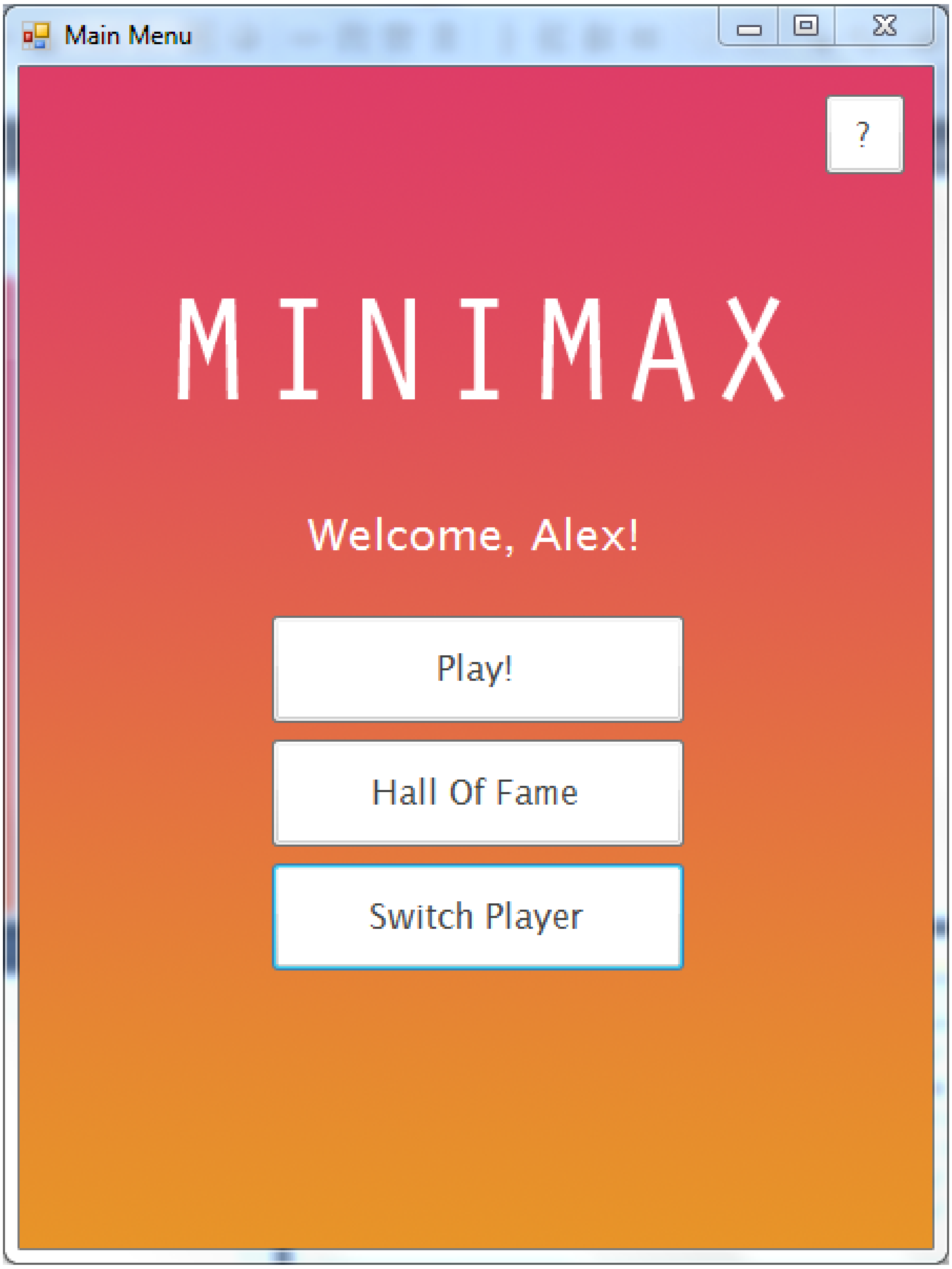
```
  BEGIN SUB btnHelp_Click(sender As Object, e As Event) When help button is clicked

    Print help message.
  END SUB
```

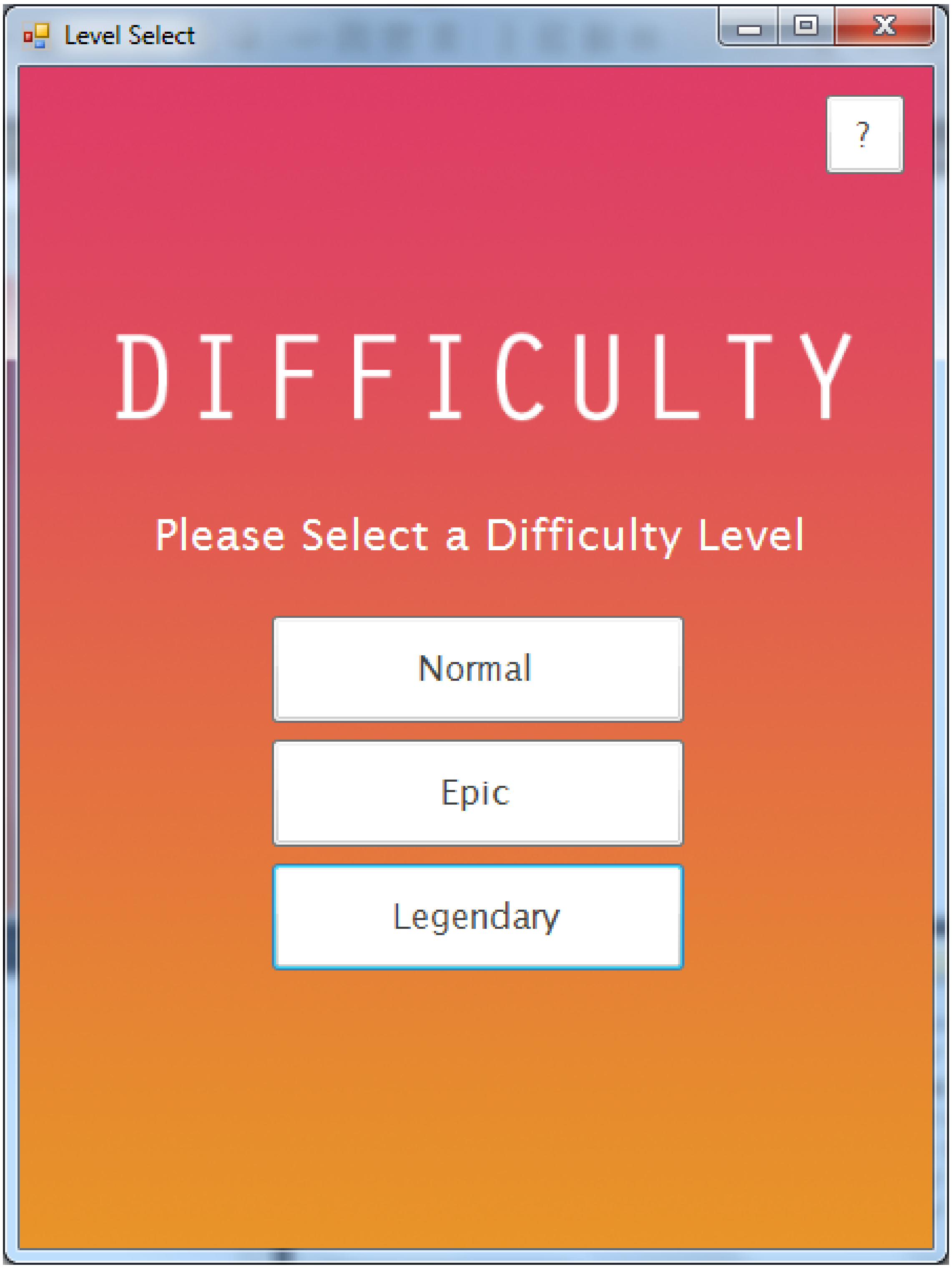
IMPLEMENTING

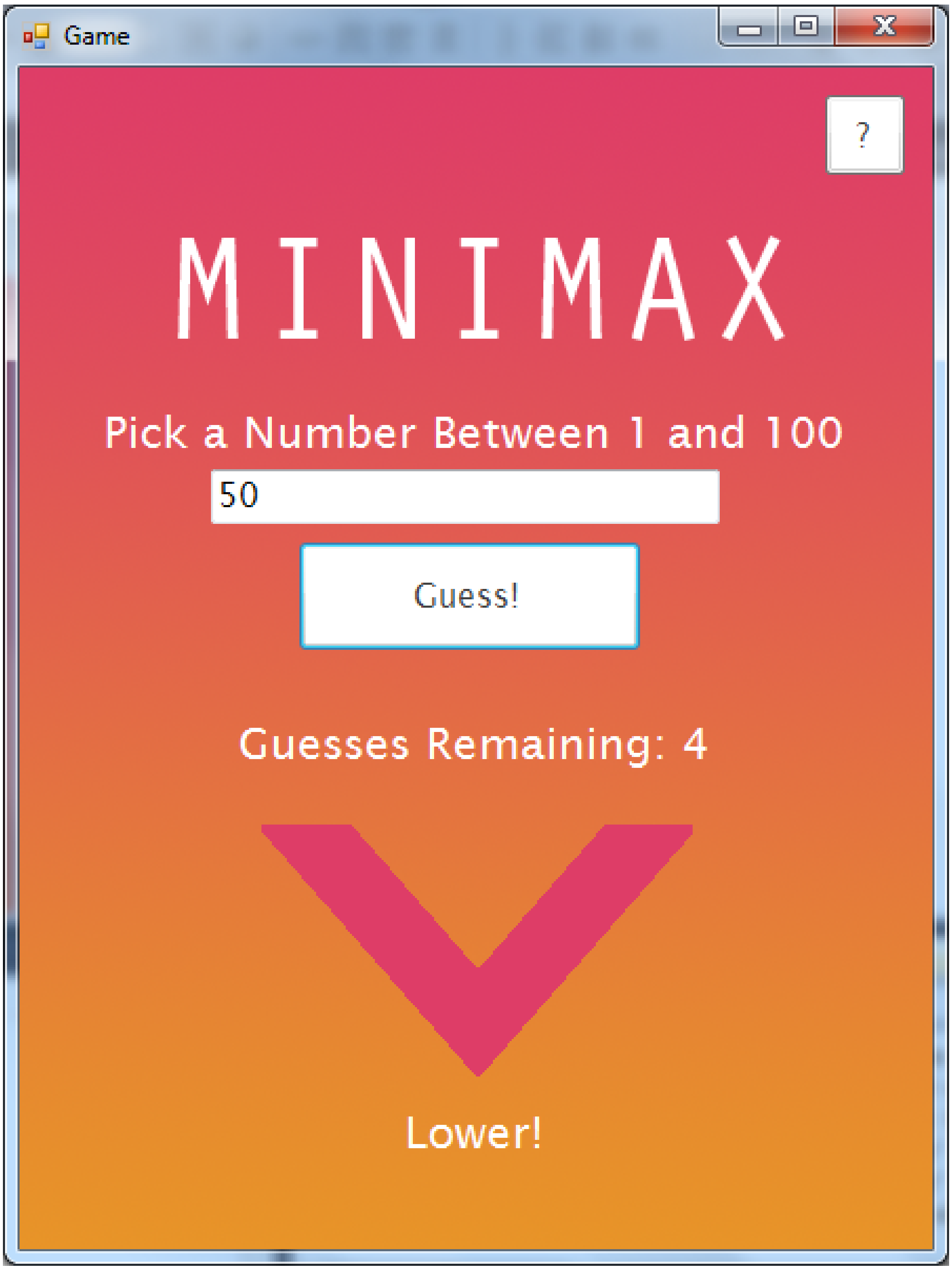
SCREENSHOTS OF FORMS

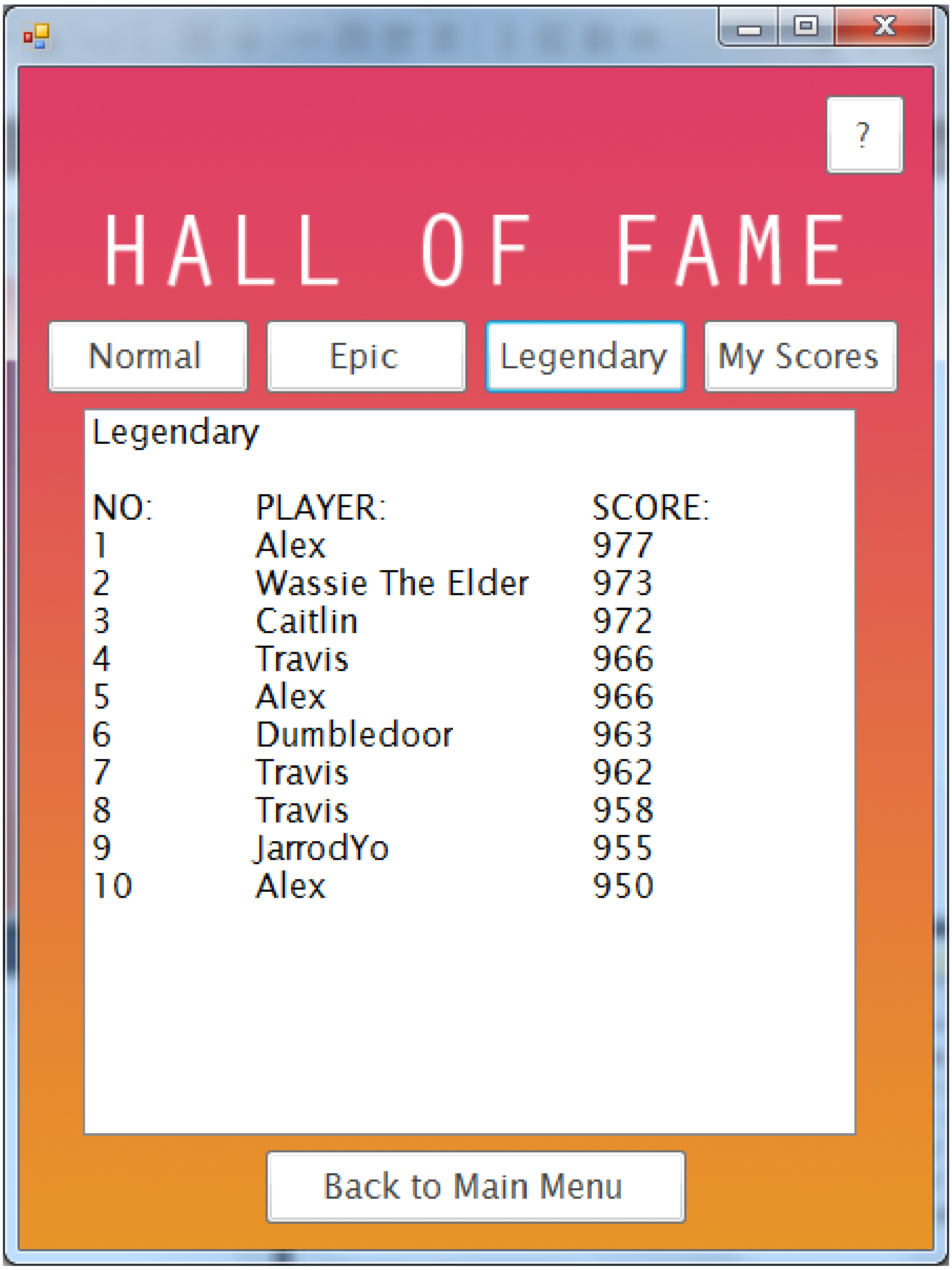



Login
?
LOGIN
Please choose a Player Name
Alex
Clear
Submit
Quit Game


Main Menu
?
MINIMAX
Welcome, Alex!
Play!
Hall Of Fame
Switch Player


Level Select
?
DIFFICULTY
Please Select a Difficulty Level
Normal
Epic
Legendary


Game
?
MINIMAX
Pick a Number Between 1 and 100
50
Guess!
Guesses Remaining: 4
Lower!



```
Legendary

NO:     PLAYER:             SCORE:
1       Alex                977
2       Wassie The Elder    973
3       Caitlin             972
4       Travis              966
5       Alex                966
6       Dumbledoor          963
7       Travis              962
8       Travis              958
9       JarrodYo            955
10      Alex                950
```

SOURCE CODE

PublicModule

```
Module PublicModule
    'For declaring public variables to be used elsewhere
    Public playerName As String 'The player's name
    Public levelDifficulty As String 'The difficulty level of the next game to be played,
determined by user.
End Module
```

ReadWriteData

```
Imports System.IO

Module ReadWriteData
    'Define new ScoreRecord structure for keeping track of scores and players, etc.
    Public Structure ScoreRecord
        Dim scoreID As Integer 'ID for reference
        Dim difficulty As String 'The difficulty level that the game was played on
        Dim playerName As String 'the player who set the score
        Dim score As Integer 'The score for the game
    End Structure

    Public arrScores(0) As ScoreRecord 'List of all data from previously won games.
    Public filePath As String = Application.StartupPath 'The file path of the project

    Public Sub LoadScores()
        'Loads order data from text file and places it in the "arrScores" array
        arrScores = {} 'reset arrScores, in case LoadScores() has previously been called
        Dim oRead As System.IO.StreamReader 'For sequentially accessing the contents of text
files.
        Dim lineIn As String 'The line read by oRead that is then processed.
        Dim tmp 'Temporary variable for storing data while it is processed and placed into
arrScores.

        oRead = File.OpenText(filePath & "\scores.txt") 'Locate text file

        While oRead.Peek <> -1 'while there are still lines left to read
            lineIn = oRead.ReadLine() 'Set lineIn to current line
            If Mid(lineIn, 1, 1) <> "" Then 'if there are characters on the line
                tmp = Split(lineIn, "|") 'Split the line, separating the elements by the "|"
character
                ReDim Preserve arrScores(UBound(arrScores) + 1) 'Open up a new element in
arrScores array
                'Assign to the element in arrScores that we just created
                arrScores(CInt(tmp(0))).scoreID = tmp(0)
                arrScores(CInt(tmp(0))).difficulty = tmp(1)
                arrScores(CInt(tmp(0))).playerName = tmp(2)
                arrScores(CInt(tmp(0))).score = tmp(3)
            End If
        End While
        oRead.Close()
    End Sub

    Private Function FindNextID() 'sets id for next record +1
        Dim x As Integer = 0 'The previous last ID in arrScores
        For i = 0 To UBound(arrScores)
            If arrScores(i).scoreID > x Then
                x = arrScores(i).scoreID
            End If
        Next
        FindNextID = x + 1
    End Function
    Public Sub CreateScoreRecord(ByVal difficulty As String, ByVal playerName As String, ByVal 
score As Integer)
        Dim NextID As Integer = FindNextID() 'The next ID in the array arrScores that we want to 
assign to.
        ReDim Preserve arrScores(NextID) 'opens up another element at end of array, ready for 
accepting next record.
        'assign to this new element of the array that we just created.
        arrScores(NextID).scoreID = NextID
        arrScores(NextID).difficulty = difficulty
        arrScores(NextID).playerName = playerName
        arrScores(NextID).score = score
    End Sub

    Public Sub SaveFile()
        Dim sep As Char = "|" 'The separator character between elements in the text file.
        Dim writeLine As String = "" 'The string to be written to file
        For i = 0 To UBound(arrScores)
            writeLine &= arrScores(i).scoreID & sep & arrScores(i).difficulty & sep & 
arrScores(i).playerName & sep & arrScores(i).score & vbCrLf
        Next
        'override the text file
        File.WriteAllText(filePath & "\scores.txt", writeLine)
    End Sub
End Module
```

FormSplash

```
Public Class formSplash
    Private Sub StartGame(sender As System.Object, e As System.EventArgs) Handles btnStart.Click
        formLogin.Show()
        Me.Hide()
    End Sub
End Class
```

FormLogin

```
Public Class formLogin
    Private Sub SubmitPlayername(sender As System.Object, e As System.EventArgs) Handles 
btnSubmitPlayername.Click
        If txtInputPlayername.Text = "" Then 'i.e. nothing in the text box
            MsgBox("Please Enter a Playername!")
        ElseIf txtInputPlayername.Text.Length > 16 Then 'i.e. playername too long
            MsgBox("Playername must be less than or equal to 16 characters long")
        Else
            playerName = txtInputPlayername.Text 'set playerName variable to what's in the 
textbox
            txtInputPlayername.Text = "" 'Reset textbox
            formMain.Show()
            Me.Close()
        End If
    End Sub

    Private Sub ClearPlayername(sender As System.Object, e As System.EventArgs) Handles 
btnClearPlayername.Click
        txtInputPlayername.Text = "" 'Reset textbox
    End Sub

    Private Sub btnQuit_Click(sender As System.Object, e As System.EventArgs) Handles 
btnQuit.Click
        End
    End Sub

    Private Sub btnHelp_Click(sender As System.Object, e As System.EventArgs) Handles 
btnHelp.Click
```

```
        MsgBox("The playername that you choose is the one that will be displayed in the Hall of
Fame. Choosing the same playername each time you play will also allow you to see all of your
previous scores in the Hall of Fame.")
    End Sub
End Class
```

FormMain

```
Public Class formMain
    Private Sub FormMainLoadEvent(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load
        lblWelcome.Text = "Welcome, " & playerName & "!" 'Display welcome message
    End Sub
    Private Sub btnPlay_Click(sender As System.Object, e As System.EventArgs) Handles
btnPlay.Click
        formLevelSelect.Show()
        Me.Close()
    End Sub

    Private Sub btnHallOfFame_Click(sender As System.Object, e As System.EventArgs) Handles
btnHallOfFame.Click
        formHallOfFame.Show()
        Me.Close()
    End Sub

    Private Sub btnSwitchPlayer_Click(sender As System.Object, e As System.EventArgs) Handles
btnSwitchPlayer.Click
        formLogin.Show()
        Me.Close()
    End Sub

    Private Sub btnHelp_Click(sender As System.Object, e As System.EventArgs) Handles
btnHelp.Click
        MsgBox("Press 'Play' to play Minimax." & (Chr(13)) &
               "Press 'Hall of Fame' to view highscores." & (Chr(13)) &
               "Press 'Switch User' to change your playername.")
    End Sub
End Class
```

FormLevelSelect

```
Public Class formLevelSelect
    Private Sub LevelSelect(sender As System.Object, e As System.EventArgs) Handles
btnNormal.Click, btnEpic.Click, btnLegendary.Click
        levelDifficulty = sender.text 'Note that levelDifficulty will always be either "Normal",
"Epic" or "Legendary"
        formGame.Show()
        Me.Close()
    End Sub

    Private Sub btnHelp_Click(sender As System.Object, e As System.EventArgs) Handles
btnHelp.Click
        MsgBox("Changing the difficulty level changes the range of numbers and the number of
guesses you are allowed:" & (Chr(13)) & (Chr(13)) &
               "Normal: Numbers between 1 and 10, 3 guesses." & (Chr(13)) &
               "Epic: Numbers between 1 and 100, 5 guesses." & (Chr(13)) &
               "Legendary: Numbers between 1 and 1000, 9 guesses.")
    End Sub
End Class
```

FormGame

```
Public Class formGame
    Private maxNum As Integer 'Maximum random number that could be generated by the program.
Determined by levelDifficulty
```

```
    Private maxGuesses As Integer 'Maximum number of guesses, determined by levelDifficulty
    Private guessesRemaining As Integer 'Guesses remaining for the current game
    Private randomNum As Integer 'Random number produced by program
    Private playerGuess As Integer 'Guess entered by player
    Private score As Integer 'Score generated by game

    Private Sub formGame_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        'Determine maxGuesses and maxNum from levelDifficulty
        If levelDifficulty = "Normal" Then
            maxNum = 10
            maxGuesses = 3
        ElseIf levelDifficulty = "Epic" Then
            maxNum = 100
            maxGuesses = 5
        ElseIf levelDifficulty = "Legendary" Then
            maxNum = 1000
            maxGuesses = 9
        End If
        guessesRemaining = maxGuesses
        'Update labels
        lblGuessesRemaining.Text = "Guesses Remaining: " & guessesRemaining
        lblGameInstructions.Text = "Pick a Number Between 1 and " & maxNum
        lblFeedback.Text = ""
        'Generate a new random number
        randomNum = GenerateRandomNum(1, maxNum)
        'Set start score to maxNum
        score = maxNum
        'Load scores into RAM in ReadWriteData.vb
        LoadScores()
    End Sub

    Private Function GenerateRandomNum(lowerBound As Integer, upperBound As Integer)
        'Generate a new random number
        Randomize()
        Dim temp As Integer = CInt(Math.Floor((upperBound - lowerBound + 1) * Rnd())) +
lowerBound 'Code from https://msdn.microsoft.com/en-us/library/f7s023d2%28v=vs.90%29.aspx,
accessed 4.7.2015. temp is a Temporary variable for storing the randomly generated number.
        Return temp
    End Function

    Private Sub btnGuess_Click(sender As System.Object, e As System.EventArgs) Handles
btnGuess.Click
        Dim guessOK As Boolean = IsValidGuess() 'Flag that is set to true when the guess has been
validated.
        If guessOK = True Then
            UpdateScore()
            CompareGuess()
        End If
    End Sub

    Private Function IsValidGuess()
        'Check that the number in the text box is a positive integer between 1 and maxNum
        Try
            playerGuess = CInt(txtInputGuess.Text)
            If playerGuess < 1 Or playerGuess > maxNum Then
                MsgBox("Please insert a number between 1 and " & maxNum)
                txtInputGuess.Text = ""
                Return False
            Else
                Return True
            End If
        Catch ex As Exception
            MsgBox("Please insert a number between 1 and " & maxNum)
            txtInputGuess.Text = ""
            Return False
        End Try
    End Function

    Private Sub CompareGuess()
        'Compare Player's guess to the number generated.
        If playerGuess > randomNum Then
            lblFeedback.Text = "Lower!"
            picArrowDown.Visible = True
            picArrowUp.Visible = False
            UpdateRemainingGuesses()
        ElseIf playerGuess < randomNum Then
            lblFeedback.Text = "Higher!"
            picArrowUp.Visible = True
            picArrowDown.Visible = False
            UpdateRemainingGuesses()
        Else
            lblFeedback.Text = "You Win!"
            picArrowUp.Visible = False
            picArrowDown.Visible = False
            'Write a new score record to file.
            CreateScoreRecord(levelDifficulty, playerName, score)
            SaveFile()
            'Display "You Win" message
            MsgBox("You Win! Your score was: " & score & "! Click OK to proceed to Hall of
Fame.")
            formHallOfFame.Show()
            Me.Close()
        End If
    End Sub

    Private Sub UpdateScore()
        score = score - Math.Ceiling(Math.Abs((playerGuess - randomNum) / (maxGuesses - 1)))
'Player loses less score if they are closer to the target.
        'Note that minimum score occurs when randomNum = maxNum and player guesses 1 for all
guesses except for the last guess, which they guess correctly.
        'This is a very rare case, but it means that 0 <= score <= maxNum always
    End Sub

    Private Sub UpdateRemainingGuesses()
        'Check if there are guesses left
        If guessesRemaining > 0 Then
            guessesRemaining = guessesRemaining - 1 'subtract 1 from remaining guesses
            lblGuessesRemaining.Text = "Guesses Remaining: " & guessesRemaining 'update label
            If guessesRemaining = 0 Then
                'Game Over
                lblFeedback.Text = "Out of Guesses!"
                picArrowUp.Visible = False
                picArrowDown.Visible = False
                MsgBox("Out of Guesses! The number was " & randomNum & "! Click OK to return to
the Main Menu.")
                formMain.Show()
                Me.Close()
            End If
        End If
    End Sub

    Private Sub btnHelp_Click(sender As System.Object, e As System.EventArgs) Handles
btnHelp.Click
        MsgBox("Pick a number between 1 and " & maxNum & ", and enter that number in the input
box." & (Chr(13)) &
               "Press 'Guess!' to confirm your guess. If you guess correctly, you win the game."
& (Chr(13)) &
               "If you guess incorrectly, you will be told whether your guess is higher or lower
than the random number." & (Chr(13)) &
               "You lose the game when you run out of guesses." & (Chr(13)) & (Chr(13)) &
               "Good Luck!" & (Chr(13)) & (Chr(13)) &
            "Extra Tip: The closer each guess is to the random number and the fewer guesses
you use, the better your score will be.")
    End Sub
End Class
```

FormHallOfFame

```
Public Class formHallOfFame
    Private arrRecordsToDisplay(0) As ScoreRecord 'List of data from previously won games,
filtered/sorted appropriately.

    Private Sub btnBackToMain_Click(sender As System.Object, e As System.EventArgs) Handles
btnBackToMain.Click
        formMain.Show()
        Me.Close()
    End Sub

    Private Sub formHallOfFame_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load
        'Show scores for the current player
        LoadScores()
        arrRecordsToDisplay = arrScores
        arrRecordsToDisplay = SortScores(arrRecordsToDisplay)
        arrRecordsToDisplay = FilterScoresByCurrentPlayer(arrRecordsToDisplay)
        DisplayScoresByPlayer(arrRecordsToDisplay)
    End Sub

    Private Sub ScoreButtonClick(sender As System.Object, e As System.EventArgs) Handles
btnNormalScores.Click, btnEpicScores.Click, btnLegendaryScores.Click, btnMyScores.Click
        LoadScores() 'Load scores from text file into RAM
        arrRecordsToDisplay = arrScores 'copy values from public arrScores into private
arrRecordsToDisplay before further processing to avoid corrupting permanent data.
        arrRecordsToDisplay = SortScores(arrRecordsToDisplay) 'Sort records by the score value,
from highest to lowest.

        If sender.tag = 3 Then
            'btnMyScores was clicked
            arrRecordsToDisplay = FilterScoresByCurrentPlayer(arrRecordsToDisplay)
            DisplayScoresByPlayer(arrRecordsToDisplay)
        Else
            'btnNormalScores, btnEpicScores or btnLegendaryScores was clicked
            arrRecordsToDisplay = FilterScoresByDifficulty(arrRecordsToDisplay, sender.text)
'Note that sender.text returns either "Normal", "Epic" or "Legendary"
            DisplayScoresByDifficulty(arrRecordsToDisplay, sender.text) 'Note that sender.text
returns either "Normal", "Epic" or "Legendary"
        End If
    End Sub

    Private Function SortScores(arrItems() As ScoreRecord)
        'Sort all scores (arrScores) from highest to lowest.
        Dim firstIndex As Integer = 0 'The first index to start the sort from
        Dim lastIndex As Integer = UBound(arrItems) 'The last index in the array to be sorted
        Dim endUnsortedIndex As Integer = firstIndex 'The number of items in the array that have
already been sorted
        Dim currentIndex As Integer 'The index of the array element that is currently being
sorted
        Dim maxItem As ScoreRecord 'The scoreRecord with the maximum score value
        Dim maxIndex As Integer 'The index of the scoreRecord with the maximum score value.
        Dim temp As ScoreRecord 'Temporary variable for holding data while elements in the array
are reshuffled.

        While endUnsortedIndex < lastIndex
            currentIndex = endUnsortedIndex
            maxItem = arrItems(currentIndex)
            maxIndex = currentIndex
            While currentIndex < lastIndex
```

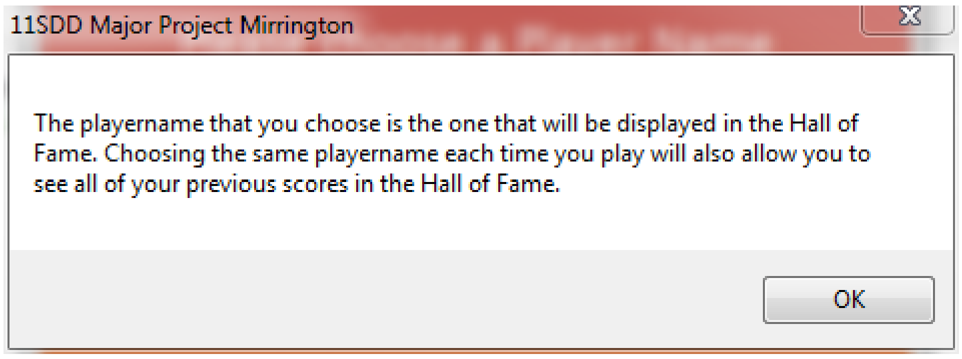
```
                currentIndex = currentIndex + 1
                If arrItems(currentIndex).score > maxItem.score Then
                    maxItem = arrItems(currentIndex)
                    maxIndex = currentIndex
                End If
            End While
            'swap items
            temp = arrItems(maxIndex)
            arrItems(maxIndex) = arrItems(endUnsortedIndex)
            arrItems(endUnsortedIndex) = temp
            endUnsortedIndex = endUnsortedIndex + 1
        End While
        Return arrItems
    End Function

    Private Function FilterScoresByCurrentPlayer(arrScoresToFilter() As ScoreRecord)
        'Filter the records by the current player
        Dim arrToReturn(0) As ScoreRecord 'The array to be returned after processing

        For i = 0 To UBound(arrScoresToFilter)
            If arrScoresToFilter(i).playerName = playerName Then
                Dim newID As Integer = UBound(arrToReturn) + 1
                ReDim Preserve arrToReturn(newID)
                arrToReturn(newID) = arrScoresToFilter(i)
            End If
        Next
        Return arrToReturn
    End Function

    Private Function FilterScoresByDifficulty(arrScoresToFilter() As ScoreRecord,
filterDifficulty As String)
        'Note that for this function to return anything, filterDifficulty must be either
"Normal", "Epic" or "Legendary"
        Dim arrToReturn(0) As ScoreRecord 'The array to be returned after processing
        For i = 0 To UBound(arrScoresToFilter)
            If arrScoresToFilter(i).difficulty = filterDifficulty Then
                Dim newID As Integer = UBound(arrToReturn) + 1 'The next ID in the array
arrToReturn that we want to assign to.
                ReDim Preserve arrToReturn(newID)
                arrToReturn(newID) = arrScoresToFilter(i)
            End If
        Next
        Return arrToReturn
    End Function

    Private Sub DisplayScoresByDifficulty(arrToDisplay() As ScoreRecord, difficulty As String)
        lstScores.Items.Clear()
        lstScores.Items.Add(difficulty) 'Display the difficulty that has been selected
        lstScores.Items.Add("")
        lstScores.Items.Add("NO:" & vbTab & "PLAYER:" & vbTab & vbTab & "SCORE:")
        If UBound(arrToDisplay) <= 10 Then
            'i.e. we want to display all scores in this case
            For i = 1 To UBound(arrToDisplay) 'Note that we start from 1 as the first element in
the array is always null, the useful data comes after that
                If arrToDisplay(i).playerName.Length < 9 Then
                    'We need two vbTab characters to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab & vbTab &
arrToDisplay(i).score)
                Else
                    'We need one vbTab character to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab &
arrToDisplay(i).score)
                End If
            Next
        Else
            'i.e. we only want to display the top 10 scores
            For i = 1 To 10 'Note that we start from 1 as the first element in the array is
always null, the useful data comes after that
                If arrToDisplay(i).playerName.Length < 9 Then
                    'We need two vbTab characters to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab & vbTab &
arrToDisplay(i).score)
                Else
                    'We need one vbTab character to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab &
arrToDisplay(i).score)
                End If
            Next
        End If

    End Sub

    Private Sub DisplayScoresByPlayer(arrToDisplay() As ScoreRecord)
        lstScores.Items.Clear()
        lstScores.Items.Add("My Scores")
        lstScores.Items.Add("")
        If UBound(arrToDisplay) = 0 Then
            'i.e. no records to display
            lstScores.Items.Add("You have not set any highscores yet!")
        Else
            lstScores.Items.Add("NO:" & vbTab & "DIFFICULTY:" & vbTab & "SCORE:")
            For i = 1 To UBound(arrToDisplay) 'Note that we start from 1 as the first element in
the array is always null, the useful data comes after that
                If arrToDisplay(i).difficulty.Length < 9 Then
                    'We need two vbTab characters to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).difficulty & vbTab & vbTab &
arrToDisplay(i).score)
                Else
                    'We need one vbTab character to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).difficulty & vbTab &
arrToDisplay(i).score)
                End If
            Next
        End If
    End Sub

    Private Sub btnHelp_Click(sender As System.Object, e As System.EventArgs) Handles
btnHelp.Click
        MsgBox("The buttons at the top of the screen sort the highscores according to difficulty
level or the current player." & (Chr(13)) &
               "Press 'Normal', 'Epic' or 'Legendary' to view the Top 10 scores for that
difficulty level." & (Chr(13)) &
               "Press 'My Scores' to view all of the previous scores for the player you are
logged in as." & (Chr(13)) &
               "Press 'Back To Main Menu' to return to the main menu.")
    End Sub
End Class
```

ONLINE HELP SYSTEM

In my project, I implemented help buttons in the top right corner of each form, and when clicked they open a message box, informing the player on what each function/button on the form does. (Note that there is no help for the splash screen, as there is only one button which is self-explanatory)

Here are the message boxes for each of my forms:

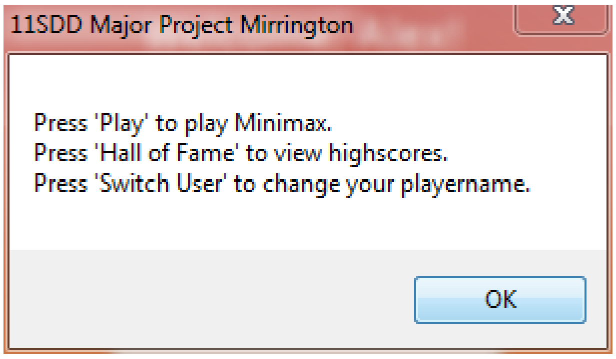
LOGIN



11SDD Major Project Mirrington

The playername that you choose is the one that will be displayed in the Hall of Fame. Choosing the same playername each time you play will also allow you to see all of your previous scores in the Hall of Fame.

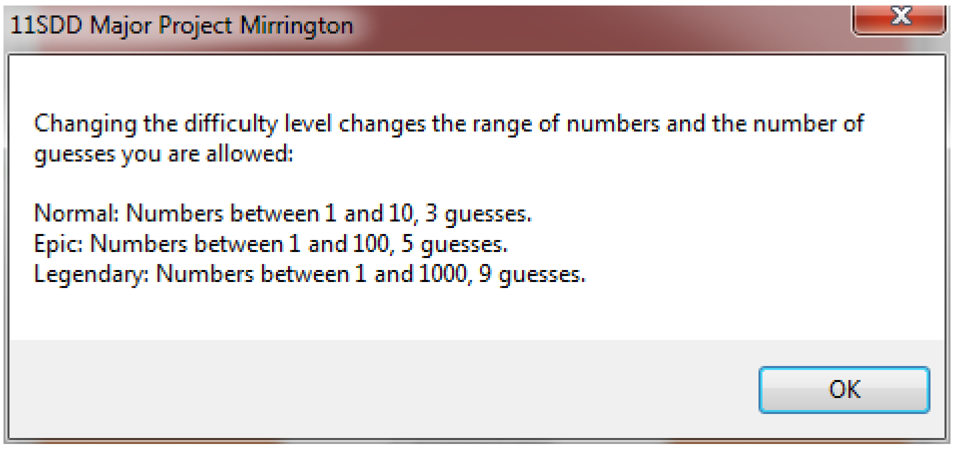
MAIN



11SDD Major Project Mirrington

Press 'Play' to play Minimax.
Press 'Hall of Fame' to view highscores.
Press 'Switch User' to change your playername.

DIFFICULTY SELECTION

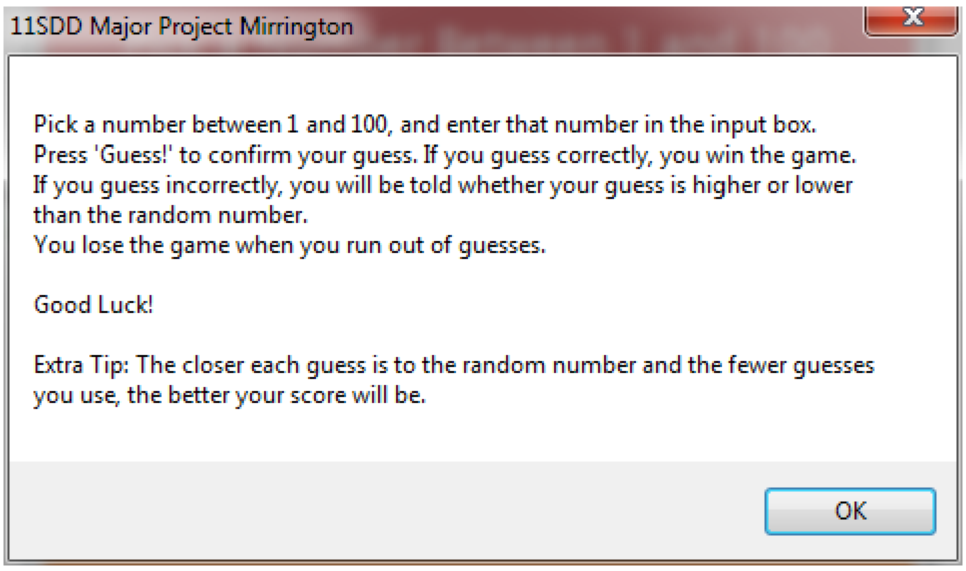


11SDD Major Project Mirrington

Changing the difficulty level changes the range of numbers and the number of guesses you are allowed:

Normal: Numbers between 1 and 10, 3 guesses.
Epic: Numbers between 1 and 100, 5 guesses.
Legendary: Numbers between 1 and 1000, 9 guesses.

GAME

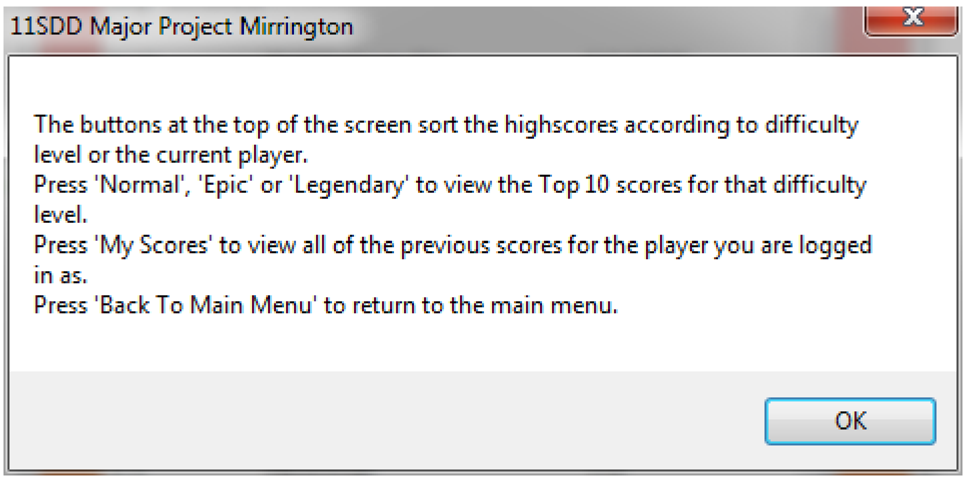


Pick a number between 1 and 100, and enter that number in the input box.
Press 'Guess!' to confirm your guess. If you guess correctly, you win the game.
If you guess incorrectly, you will be told whether your guess is higher or lower than the random number.
You lose the game when you run out of guesses.

Good Luck!

Extra Tip: The closer each guess is to the random number and the fewer guesses you use, the better your score will be.

HALL OF FAME



The buttons at the top of the screen sort the highscores according to difficulty level or the current player.
Press 'Normal', 'Epic' or 'Legendary' to view the Top 10 scores for that difficulty level.
Press 'My Scores' to view all of the previous scores for the player you are logged in as.
Press 'Back To Main Menu' to return to the main menu.

TESTING + EVALUATING

TEST DATA TABLE

TEST DATA	REASON FOR TEST	EXPECTED OUTPUT	RESULT
SPLASH SCREEN			
Press ‘Press To Start’ button	Make sure that the transition between Splash screen and Login screen occurs correctly.	Splash screen closes and Login form loads on button press.	Pass
LOGIN SCREEN			
Press ‘Quit Game’ button	Make sure that the game quits correctly.	Program ends.	Pass
Press ‘Clear’ button	Make sure the text box clears correctly	Clear input text box	Pass
Null value for the string in the input box	Make sure that the string in the input box is not accepted as a value for the playerName variable.	Message box displayed, informing user of the error.	Pass
“0123456789abcdefg” passed into the input box	Make sure that the string in the input box is not accepted as a value for the playerName variable, as it is too long (>16 characters long)	Message box displayed, informing user of the error.	Pass
“Alex” passed into the input box	Make sure that the string in the input box is accepted as a value for the playerName variable, and the main form loads. The playerName is used in the Hall of Fame, as well as displayed on the main menu.	playerName set to “Alex” Main menu loads, login form closes.	Pass
MAIN MENU			
Press ‘Play!’ button	Make sure that the transition between Main menu and Difficulty form occurs correctly.	Main menu closes and Difficulty selection form loads on button press.	Pass
Press ‘Hall Of Fame’ button	Make sure that the transition ~~between Main menu and Hall of~~ Fame screen occurs correctly.	Main menu closes and Hall ~~of Fame form loads on~~ button press.	~~Pass~~
Press ‘Switch Player’ button	Make sure that the transition between Main menu and Login screen occurs correctly.	Main menu closes and Login form loads on button press.	Pass
LEVEL SELECT			
Press ‘Normal’, ‘Epic’ or ‘Legendary’ button	Make sure that the appropriate difficulty is selected and that the game is able to use it.	Update public variable levelDifficulty to the name of the button.	Pass
GAME SCREEN			
“50” passed into the	Make sure that the game	Update image and feedback	Pass

input box	provides appropriate feedback to the user if the guessed number is higher than the random number (in this case, ranNum = 37)	label to indicate 'Higher', subtract 1 from guessesRemaining	
"20" passed into the input box	Make sure that the game provides appropriate feedback to the user if the guessed number is lower than the random number (in this case, ranNum = 37)	Update image and feedback label to indicate 'Lower, subtract 1 from guessesRemaining	Pass
"37" passed into the input box (correct number)	Make sure that the game recognises that the user won	Message box displayed, informing user of their win and final score, load hall of fame and close game form. Also save the score Record to file.	Pass
"-5" passed into the input box	Make sure that the game doesn't accept this as an input, as it is out of range (range in this case is 1-100)	Message box displayed, informing user of the error.	Pass
"501" passed into the input box	Make sure that the game doesn't accept this as an input, as it is out of range (range in this case is 1-100)	Message box displayed, informing user of the error.	Pass
"a3b" passed into the input box	Make sure that the game doesn't accept this as an input, as it contains letters.	Message box displayed, informing user of the error.	Pass
Nothing passed into the input box	Make sure that the game doesn't accept this as an input, as it doesn't contain a guess	Message box displayed, informing user of the error.	Pass
HALL OF FAME			
Press 'Normal', 'Epic' or 'Legendary' button	Make sure that the scores are retrieved properly from file and sorted, filtered and displayed properly.	Retrieve scores from file, sort them highest to lowest by score, filter out only the top 10 scores for the difficulty that was pressed e.g. 'Epic' and display these scores in the list box.	Pass
Press 'My Scores' button	Make sure that the scores are retrieved properly from file and sorted, filtered and displayed properly.	Retrieve scores from file, sort them highest to lowest by score, filter out all scores for the current player and display these in the list box.	Pass
Press 'Back to Main Menu' button	Make sure that the transition between Hall of Fame and Main menu screen occurs correctly.	Main menu opens and Hall of Fame form closes on button press.	Pass
GENERAL			
Press '?' help button	Make sure that the appropriate online feedback for the form is displayed.	Display appropriate help message box to the user.	Pass

EXTENDED RESPONSES

Intellectual Property

Who 'owns' the rights to the software developed? Why? What would be the most appropriate type of license for this program?

The rights to the software that I have developed belong to GameTech, as they approached me to undertake the project. In a real world scenario, a formal agreement would be made between the contracting company and the developer before work begins on a project in order to clearly define who the rights of the software belong to. The most likely terms of agreement would be that all resources developed belong to the contracting company. If I decided not to agree to these restrictions before working on the project, then Gametech would simply find someone else to complete the project.

I would not be able to use any code that I have developed in my own personal project in the future, as the intellectual property of the project belongs to GameTech, not to me, and I would thus be breaching intellectual property laws. If I was to continue working for the same company, however, I would most likely be able to reuse modules of code for future projects for the company.

The most appropriate license for the program would be a commercial software license, as the company (GameTech) aims to sell the software in a package called "PC Fun Zone". The commercial software license is covered by copyright law, so the package can't be redistributed by end users. It is in GameTech's interests to ensure that the software can't be modified or reverse-engineered, due to the investment that the company has made in developing the product. Source code would not be distributed to end users, in order to protect the intellectual property of the company.

Privacy

Does the program adequately protect the information it stores? Is it necessary to make improvements on data protection? If so, what could be done?

In its current, uncompiled form, the program allows easy access to the information stored within it. The data stored by the program is unencrypted, and so is easily manipulated or read. The data could be protected more by properly compiling the application and implementing encryption/decryption algorithms. However, this is not entirely necessary for the scope of my project, as it contains no personal information that could breach someone's rights of privacy if the data was misused; In my program, the most that could occur in terms of manipulation of data would be someone creating a new high score in the text file.

However, if my project did contain sensitive personal information such as people's full names, addresses, phone numbers or credit card details, protecting this information using methods such as encryption would be far more important. Whenever storing and processing personal information, software developers have to ensure the user that their information is secure, explain how it will be used and explain the purpose for collecting the information.

Ergonomics Issues

What ergonomic factors need to be considered in the design of this program?

Ergonomics refers to the ease with which humans are able to function within a particular environment, for example, a work environment. As the software developer, it is in my interests to make my product as ergonomically sound as possible, to ensure that it is accepted and enjoyed by the end user as a form of entertainment.

When designing my application, it was important to maintain consistency throughout the project. By designing all of my user interfaces with similar colour schemes, fonts, appropriate user interface elements and adequate blank space, I was able to ensure that the user is able to navigate between different screens efficiently and essentially makes the whole application more intuitive and more likely to be accepted by the end user; A consistent colour scheme and similar fonts across forms make the application feel less disjointed and allow the end user to familiarize themselves with the application more quickly. By using easily recognizable and generally accepted user interface elements, I was able to greatly decrease the learning curve for my application, as the end user will be able to recognize and use the functions of the program more intuitively. The use of blank space and spreading functions out over various forms makes the program more intuitive and less daunting to the end user.

During the design phase, it was also important to consider how to provide appropriate feedback to the user as they interact with the software. The main thing that I wanted to focus on in terms of providing feedback was to make sure that the user feels in control of the application. As soon as they feel that the computer has ‘taken over’, the application becomes frustrating and unenjoyable to use. By providing simple yet intuitive feedback such as “Please insert a number between 1 and 10”, the user learns that they made an error, but still feel in control of the program, and are able to immediately recognize what they need to do to fix the problem.

Another factor that had to be considered was maintaining acceptable response times. As soon as the user has to wait longer than around a second for an action to occur, they begin to feel like the computer is in control of the program, not them. This eventually causes frustration. While some people decided to include unnecessary wait times for their splash screens, I decided to include a single button that takes you to the next screen when pressed, instead of forcing the user to wait 3 seconds. This makes the user feel more capable when using the program and makes the user experience much more enjoyable.

All of these factors are essential to include in software products in order to ensure that the user is able to interact with the product in an intuitive and efficient manner.

Inclusivity & Accessibility

What design factors will maximise inclusivity and accessibility?

A software product that is inclusive and accessible accounts for the differences in ability between various end users. By ensuring that our software products are inclusive, we are able to increase the range of end users and thus make our product more profitable. Software products can be made more inclusive by including:

- Support for both genders – It is important to make sure that the product is accepted by both male and female users. By seeking feedback from male and female colleagues, I was able to ensure that my program was supported by both genders, in terms of the colours, fonts and layout of the project. Including support for both genders rather than one essentially doubles the range of possible end users, thus maximizing inclusivity.
- Support for disabilities – Disabilities come in many different forms (such as visual and auditory) and are thus hard to cater for in a software solution. However some general techniques can be employed in order to maximise inclusivity/accessibility for disabled users; In my project for example, I made sure that no essential user feedback relied only on sound. E.g. When the user makes a guess, the “Higher”/”Lower” feedback is an essential part of the game, and doesn’t require the person being able to hear anything in able to play the game. On the other hand, all of the feedback is displayed visually, and thus is not inclusive of people with a visual disability. A better solution would be to present both visual and audible feedback, thus including a wider range of end users.
- Support for multiple cultures and languages – Whilst this factor was not entirely relevant for the scope of my project, it is still an important factor to consider; By limiting your product to one language, you are greatly reducing the number of possible end users for the product. By including multiple languages and ensuring that it is not offensive towards certain groups of people, inclusivity can be maximized.
- Support for users of various economic backgrounds – Again, this was not as relevant for my project as I don’t intend to sell it, but it is an important factor to balance; Making a product that has more features might be more successful, but it costs more money to produce, and thus might not be economically viable for some users. It is important to assess the requirements of the end user in order to maximise both inclusivity/accessibility and functionality simultaneously.

By considering these aspects carefully when developing a project, we can maximise inclusivity/accessibility and thus increase the success of our product.

LEARNING JOURNAL

Blog URL:

http://alexmirrington11sdd.blogspot.com.au

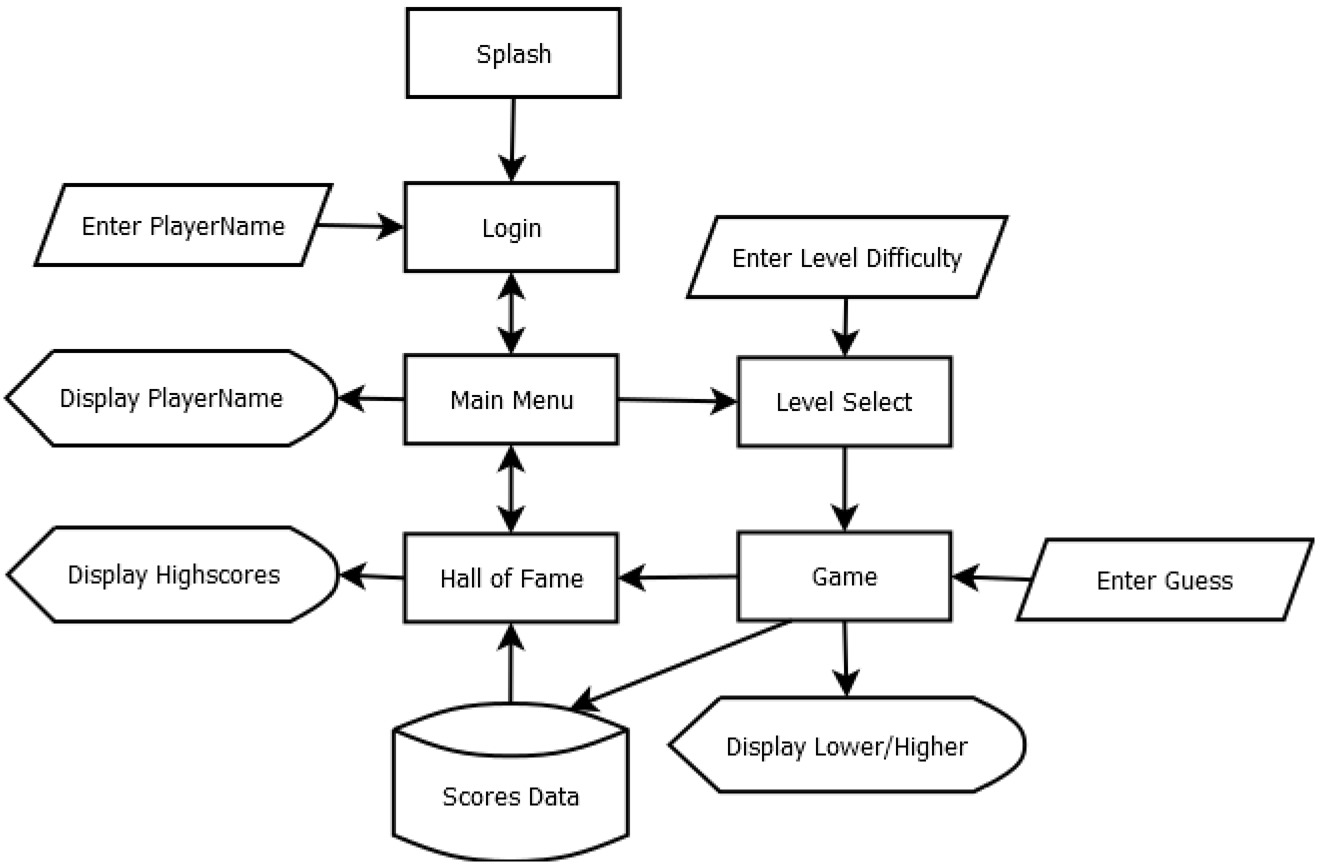
4/5/15 (Week 2)
I have completed the introductory stage of my project (the problem statement), this can be found in the previous post. I have not learnt much in regards to the project so far, as I have only been working on it for a total of a few days. I have exams during the next two weeks, so will be unable to do any project work, however in the week after exams, I plan to complete my Gantt chart as soon as possible and then begin on GUI mockups, storyboards and context diagrams.

21/5/15 (Week 5)
This week I worked on my Gantt chart and began work on the planning stage of my project. Ideally, the initial Gantt chart would have been completed in Week 2, however I ran out of time. Consequently, this was the first task I completed this week. I will create a revised version of my Gantt chart throughout the project, detailing the specific order in which tasks were undertaken, rather than the planned schedule for the project. Before beginning work on the planning and designing stage of my project, we had a few class discussions and brainstorms about the scope of the project in terms of the planning stage, and came up with some draft IPO charts, context diagrams, data flow diagrams and data dictionaries. Here is my data dictionary so far, for example:

NAME	Data Type	Scope	Description	Example	Validation	Format
playerName	String		The player's name	Alex		
levelDifficulty	String			Easy		
maxGuesses	Integer		Maximum number of guesses, determined by levelDifficulty	3		
maxNum	Integer		Maximum random number that could be generated by the program. Determined by levelDifficulty	10		
randomNum	Integer		Random number produced by program	7		
playerGuess	Integer		Guess entered by player	6		
gameDate	Date		Date that game was played	21/05/15		DD/MM/YYYY
gameScore	Integer		Score generated by game			
gameRecord	Structure/Record		Record of each win to be used in hall of fame	1 Alex 21/05/15 Easy		ID (Int) Name (String) Score (Int) Date (Date) Level (String)
arrGameRecords	Array (latestGameRecord)		List of all data from previously won games.			

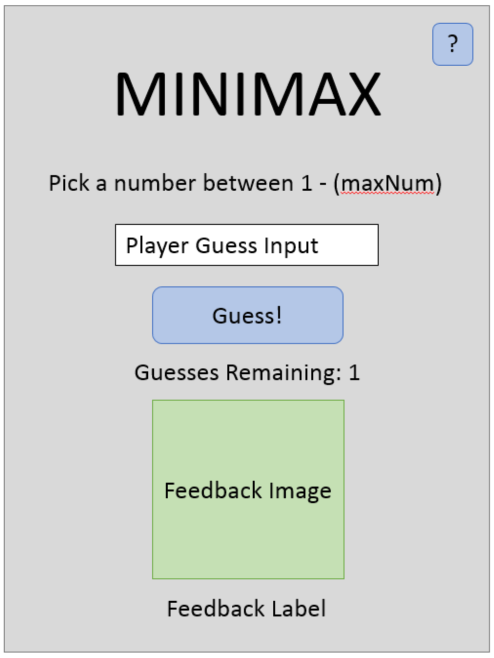
Whilst these elements of the planning stage need to be refined and tailored to fit my own specifications for the project, I have a reasonable starting point for the planning stage of my project. Next week, I will work on refining each of these planning stages covered in class discussions, as well as start my GUI sketches, storyboard and system flow chart.

29/5/15 (Week 6)
I started this week by refining my IPO Chart, Data Dictionary and Data Flow Diagram, adding in more detailed elements that are more specific to my project than what was previously discovered in class. We also discussed the purpose and implementation of system flow charts as a class, and I used what I learnt to construct my own flow chart, relevant to my project:



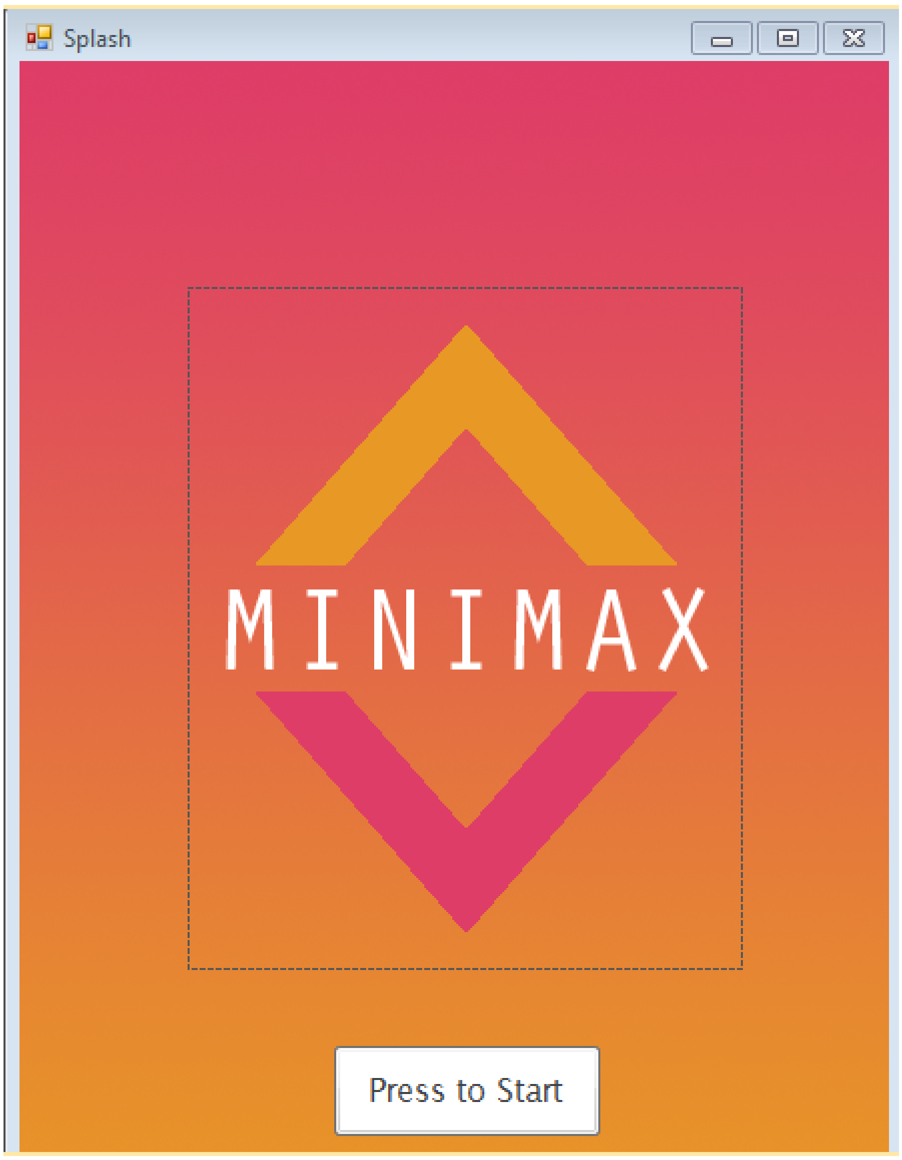
I completed my storyboard and GUI mockups this week as well. I decided to go for a 3:4 aspect ratio, similar to an iPad, as I wanted to minimize the number of GUI elements on the screen at once. I am aiming for a minimalist theme, focusing on simple colours and shapes. Here is an example of my Game form:

GAME
450x600



I didn't quite get around to my structure chart this week, but I will hopefully be able to get this done early next week, and begin work on creating forms in visual basic. I need to have started the implementing stage of my project by the start of Week 8.

5/6/15 (Week 7)
We spent a bit more of our class time on theory work this week, so I had a bit less time to work on my project. We did, however, discuss the nature and implementation of algorithm flowcharts, completing a few example flowcharts in class. I have decided to reshuffle my Gantt chart, and work on my algorithms at the same time as coding them in visual basic. Whilst this is not ideal for the structured approach, it will certainly speed up the development process, as I will immediately be able to tell whether an algorithm will work or not, and change my flowcharts and pseudocode accordingly. I began work on creating my forms in visual basic. Here is my splash screen:



I am slightly behind schedule, so will need to catch up on refining some of my planning tools at home. I am hoping to have started coding by early next week.

12/6/15 (Week 8)
I began coding this week, focusing on the navigation between forms. The main challenge I came across was regarding subroutines that run when a form loads for the first time.

```
Private Sub FormMainLoadEvent(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load
        lblWelcome.Text = "Welcome, " & playerName & "!"
End Sub
```

Above is the load event subroutine for my main menu. In the 'handles' section of the subroutine, you'll notice `MyBase.Load` – this signifies that the subroutine runs the first time the form is loaded.
I also have another subroutine in my Main form, which runs when the "Switch Player" button is pressed:

```
Private Sub btnSwitchPlayer_Click(sender As System.Object, e As
System.EventArgs) Handles btnSwitchPlayer.Click
        formLogin.Show()
        Me.Close()
    End Sub
```

I originally used `Me.Hide()` instead of `Me.Close()`. This meant that the `FormMainLoadEvent()` function didn't run when the player returned to the main menu after switching their username. I got around this problem by using `Me.Close()` so when the user reopens the main menu, the program interprets this as opening the form for the first time and thus runs the `FormMainLoadEvent()` function, updating the welcome message label.

I also learnt a bit about Try/Catch statements. Here is an example of a Try/Catch statement for the input box that the player puts their guess in:

```
Private Function IsValidGuess()
        'Check that the number in the text box is a positive integer between
1 and maxNum
        Try
            playerGuess = CInt(txtInputGuess.Text)
            If playerGuess < 1 Or playerGuess > maxNum Then
                MsgBox("Please insert a number between 1 and " & maxNum)
                txtInputGuess.Text = ""
                Return False
            Else
                Return True
            End If
        Catch ex As Exception
            MsgBox("Please insert a number between 1 and " & maxNum)
            txtInputGuess.Text = ""
            Return False
        End Try
    End Function
```

I am learning lots about the implementation stage of the development process, and realizing how difficult it is to follow your original plans to each and every detail. The reality is that things change during the implementing stage, as you become more familiar with the project.

17/6/15 (Week 9)
I implemented the logic for comparing the player's guess to the randomly generated number this week. The underlying logic is reasonably simple, and can be summed up in a simple pseudocode statement:

```
START CompareGuess()
        IF playerGuess > randomNum THEN
                Update feedback label to “Lower”
        ELSE IF playerGuess < randomNum THEN
                Update feedback label to “Higher”
        ELSE
                Update feedback label to “You Win”
                Load Hall of Fame
        END IF
END CompareGuess()
```

Here is the code as I implemented it in visual basic:

```
    Private Sub CompareGuess()
        'Compare Player's guess to the number generated.
        If playerGuess > randomNum Then
            lblFeedback.Text = "Lower!"
            UpdateRemainingGuesses()
        ElseIf playerGuess < randomNum Then
            lblFeedback.Text = "Higher!"
            UpdateRemainingGuesses()
        Else
            lblFeedback.Text = "You Win!"
            MsgBox("You Win! Your score was: " & score & "! Click OK to
proceed to Hall of Fame.")
            formHallOfFame.Show()
            Me.Close()
        End If

        'Update feedback label and image.
    End Sub
```

I also have a feedback image that I want to update, showing an up or down facing arrow depending on whether the random number is higher of lower than the player's guess. I'm not completely sure of the code involved in implementing this, but I'll do some research and work out how to do it. I currently have to implement a scoring system, and then read and write data to and from a file for use in the Hall of Fame. Next Term I will be able to finish that off and start work on the testing and maintaining sections of my portfolio.

17/7/15 (Week 1)
I have been finishing up my source code this week, which involved finalizing my game code, as well as implementing all of the code for the Hall of Fame. The Hall of Fame is really the most technically difficult part of the project, as it involves reading and writing to a text file. Here is what I came up with for reading and writing from file:

```
Imports System.IO

Module ReadWriteData
    'Define new ScoreRecord structure for keeping track of scores and players, etc.
    Public Structure ScoreRecord
        Dim scoreID As Integer
        Dim difficulty As String
        Dim playerName As String
        Dim score As Integer
    End Structure

    Public arrScores(0) As ScoreRecord
    Public filePath As String = Application.StartupPath
```

Note the structure that I have declared above. It will be able to hold the difficulty level, playerName and score for the game that was just won.

The LoadScores subroutine loads the permanent data from the text file into an array in RAM (arrScores), so that I can do something with the data; i.e. display it in the hall of fame.

```
    Public Sub LoadScores()
        'Loads order data from text file and places it in the "arrScores" array
        arrScores = {} 'reset arrScores, in case LoadScores() has previously been called
        Dim oRead As System.IO.StreamReader
        Dim lineIn As String
        Dim tmp

        oRead = File.OpenText(filePath & "\scores.txt")

        While oRead.Peek <> -1
            lineIn = oRead.ReadLine()
            If Mid(lineIn, 1, 1) <> "" Then
                tmp = Split(lineIn, "|")
                ReDim Preserve arrScores(UBound(arrScores) + 1)
                arrScores(CInt(tmp(0))).scoreID = tmp(0)
                arrScores(CInt(tmp(0))).difficulty = tmp(1)
                arrScores(CInt(tmp(0))).playerName = tmp(2)
                arrScores(CInt(tmp(0))).score = tmp(3)
            End If
        End While
        oRead.Close()
    End Sub

    Private Function FindNextID() 'sets id for next record +1
        Dim x As Integer = 0
        For i = 0 To UBound(arrScores)
            If arrScores(i).scoreID > x Then
                x = arrScores(i).scoreID
            End If
        Next
        FindNextID = x + 1
    End Function
```

The CreateScoreRecord subroutine opens a new element up in the arrScores array, and assigns the results of the previous game to this new element. (Note that this is called after the player wins a game.)

```
    Public Sub CreateScoreRecord(ByVal difficulty As String, ByVal playerName As String, ByVal
score As Integer)
        Dim NextID As Integer = FindNextID()
        ReDim Preserve arrScores(NextID) 'opens up another element at end of array, ready for
accepting next record.
        'assign to this new element of the array that we just created.
        arrScores(NextID).scoreID = NextID
        arrScores(NextID).difficulty = difficulty
        arrScores(NextID).playerName = playerName
        arrScores(NextID).score = score
    End Sub
```

The SaveFile subroutine permanently writes the contents of arrScores to file. (Note that this is called after the player wins a game, before the hall of fame is loaded.)

```
    Public Sub SaveFile()
        Dim sep As Char = "|"
        Dim writeLine As String = ""
        For i = 0 To UBound(arrScores)
            writeLine &= arrScores(i).scoreID & sep & arrScores(i).difficulty & sep &
arrScores(i).playerName & sep & arrScores(i).score & vbCrLf
        Next
        'override the text file
        File.WriteAllText(filePath & "\scores.txt", writeLine)
    End Sub
End Module
```

So what does all of this look like in practice? Here is what is contained inside the text file:

```
0|||0
1|Normal|Alex|8
2|Normal|Alex|9
3|Legendary|Wassie The Elder|946
4|Legendary|mole|913
5|Legendary|Travis|958
6|Legendary|Wassie The Elder|973
7|Normal|Alex|8
8|Normal|Alex|10
9|Normal|Alex|8
10|Normal|Alex|9
```

```
11|Legendary|JarrodYo|955
12|Epic|Alex|90
13|Legendary|Alex|938
14|Epic|pinis|90
15|Legendary|Caitlin|972
16|Epic|Alex|82
17|Epic|Alex|84
18|Legendary|Alex|950
19|Epic|Dumbledoor|91
20|Normal|Caitlin|8
21|Epic|Alex|94
22|Epic|Dumbledoor|97
23|Legendary|Dumbledoor|963
24|Legendary|Travis|966
25|Normal|Travis|9
26|Normal|Travis|8
27|Normal|Travis|9
28|Normal|Travis|7
29|Normal|Travis|9
30|Normal|Travis|8
31|Normal|z|10
32|Legendary|Alex|977
33|Legendary|Travis|962
34|Legendary|Alex|850
35|Legendary|Alex|966
36|Normal|charlotte|8
37|Normal|Alex|8
38|Normal|Alex|10
39|Legendary|seb|974
40|Epic|Alex|64
41|Normal|Alex|5
```

Each line in the text file contains the scoreID, difficulty, playerName and score for a certain game session, separated by a pipe character. When the LoadScores subroutine is called, all of this data is loaded into the arrScores array.

One of the main problems that I ran into was that I was calling the LoadScores subroutine multiple times (i.e. after each time a game is played), and the line:

```
ReDim Preserve arrScores(UBound(arrScores) + 1)
```

Was causing me some problems. The “ReDim Preserve” keywords mean that the array is just added to and not reset. Upon calling this line more than once, however, I began to get duplicates of all of my data records stored in my text file. To fix this, I added in this line at the start of the subroutine:

```
arrScores = {} 'reset arrScores, in case LoadScores() has previously been called
```

This just resets arrScores before loading in all of the data, fixing the issue with duplicate records.

Next week I will work on my online help system for the product, as well as work on sorting and searching algorithms for my hall of fame.

24/7/15 (Week 2)

The main goals of this week were to work on search and sort algorithms for the Hall of Fame part of my project, implement an online help system for the game, as well as work on a test data table. In the Hall of Fame, the scores are sorted from highest to lowest, and are then filtered by either the difficulty at which the score was set, or by the current player’s playerName. Here are the main parts behind my search and sort algorithms:

The SortScores function accepts an array of ScoreRecords as a parameter, and returns the same array except sorted by the score value of the ScoreRecord structure.

```
    Private Function SortScores(arrItems() As ScoreRecord)
        'Sort all scores from highest to lowest.
        Dim firstIndex As Integer = 0
        Dim lastIndex As Integer = UBound(arrItems)
        Dim endUnsortedIndex As Integer = firstIndex
        Dim currentIndex As Integer
        Dim maxItem As ScoreRecord
        Dim maxIndex As Integer
        Dim temp As ScoreRecord

        While endUnsortedIndex < lastIndex
            currentIndex = endUnsortedIndex
            maxItem = arrItems(currentIndex)
            maxIndex = currentIndex
            While currentIndex < lastIndex
                currentIndex = currentIndex + 1
                If arrItems(currentIndex).score > maxItem.score Then
                    maxItem = arrItems(currentIndex)
                    maxIndex = currentIndex
                End If
            End While
            'swap items
            temp = arrItems(maxIndex)
            arrItems(maxIndex) = arrItems(endUnsortedIndex)
            arrItems(endUnsortedIndex) = temp
            endUnsortedIndex = endUnsortedIndex + 1
        End While
        Return arrItems
    End Function
```

The FilterScoresByCurrentPlayer function accepts an array of ScoreRecords as a parameter, and returns only the elements in the array that have a playerName field the same as the currently playing player.

```
    Private Function FilterScoresByCurrentPlayer(arrScoresToFilter() As ScoreRecord)
        'Filter the records by the current player
        Dim arrToReturn(0) As ScoreRecord
        For i = 0 To UBound(arrScoresToFilter)
            If arrScoresToFilter(i).playerName = playerName Then
                Dim newID As Integer = UBound(arrToReturn) + 1
                ReDim Preserve arrToReturn(newID)
                arrToReturn(newID) = arrScoresToFilter(i)
            End If
        Next
        Return arrToReturn
    End Function
```

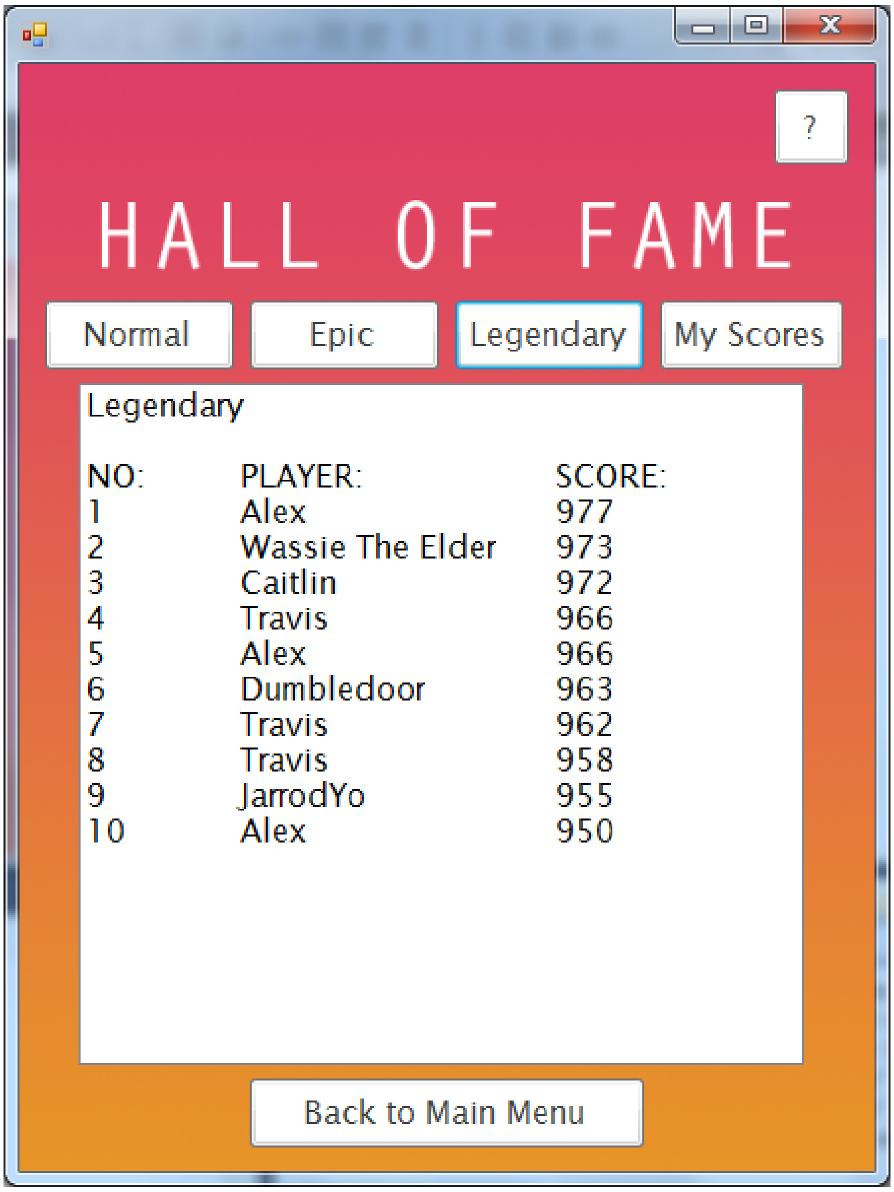
The FilterScoresByDifficulty function accepts an array of ScoreRecords and a string as parameters, and returns only the elements in the array that have a difficulty field the same as the parameter ‘filterDifficulty’ that was passed in. It should be noted that the function won’t return anything unless filterDifficulty is either “Normal”, “Epic” or “Legendary”.

```
    Private Function FilterScoresByDifficulty(arrScoresToFilter() As ScoreRecord,
filterDifficulty As String)
        'Note that for this function to return anything, filterDifficulty must be either
"Normal", "Epic" or "Legendary"
        Dim arrToReturn(0) As ScoreRecord
        For i = 0 To UBound(arrScoresToFilter)
            If arrScoresToFilter(i).difficulty = filterDifficulty Then
                Dim newID As Integer = UBound(arrToReturn) + 1
                ReDim Preserve arrToReturn(newID)
                arrToReturn(newID) = arrScoresToFilter(i)
            End If
        Next
        Return arrToReturn
    End Function
```

The next two subroutines are similar, as they both add items to the on-screen listbox. The first is for displaying scores that are filtered by difficulty (Note that it only displays the top 10), and the second is for displaying scores that are filtered by the current player.

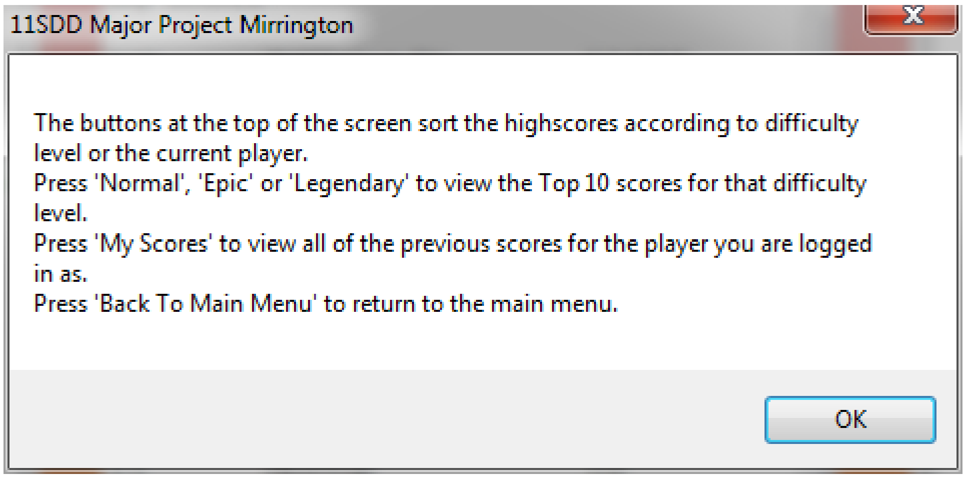
```
    Private Sub DisplayScoresByDifficulty(arrToDisplay() As ScoreRecord, difficulty As String)
        lstScores.Items.Clear()
        lstScores.Items.Add(difficulty) 'Display the difficulty that has been selected
        lstScores.Items.Add("")
        lstScores.Items.Add("NO:" & vbTab & "PLAYER:" & vbTab & vbTab & "SCORE:")
        If UBound(arrToDisplay) <= 10 Then
            'i.e. we want to display all scores in this case
            For i = 1 To UBound(arrToDisplay) 'Note that we start from 1 as the first element in
the array is always null, the useful data comes after that
                If arrToDisplay(i).playerName.Length < 9 Then
                    'We need two vbTab characters to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab & vbTab &
arrToDisplay(i).score)
                Else
                    'We need one vbTab character to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab &
arrToDisplay(i).score)
                End If
            Next
        Else
            'i.e. we only want to display the top 10 scores
            For i = 1 To 10 'Note that we start from 1 as the first element in the array is
always null, the useful data comes after that
                If arrToDisplay(i).playerName.Length < 9 Then
                    'We need two vbTab characters to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab & vbTab &
arrToDisplay(i).score)
                Else
                    'We need one vbTab character to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).playerName & vbTab &
arrToDisplay(i).score)
                End If
            Next
        End If
    End Sub

    Private Sub DisplayScoresByPlayer(arrToDisplay() As ScoreRecord)
        lstScores.Items.Clear()
        lstScores.Items.Add("My Scores")
        lstScores.Items.Add("")
        If UBound(arrToDisplay) = 0 Then
            'i.e. no records to display
            lstScores.Items.Add("You have not set any highscores yet!")
        Else
            lstScores.Items.Add("NO:" & vbTab & "DIFFICULTY:" & vbTab & "SCORE:")
            For i = 1 To UBound(arrToDisplay) 'Note that we start from 1 as the first element in
the array is always null, the useful data comes after that
                If arrToDisplay(i).difficulty.Length < 9 Then
                    'We need two vbTab characters to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).difficulty & vbTab & vbTab &
arrToDisplay(i).score)
                Else
                    'We need one vbTab character to maintain alignment
                    lstScores.Items.Add(i & vbTab & arrToDisplay(i).difficulty & vbTab &
arrToDisplay(i).score)
                End If
            Next
        End If
    End Sub
```

So what does all of this look like in practice? Here is a screenshot of the Hall of Fame form with the list box displaying the top 10 scores for the ‘Legendary’ difficulty level:



```
Legendary

NO:     PLAYER:             SCORE:
1       Alex                977
2       Wassie The Elder    973
3       Caitlin             972
4       Travis              966
5       Alex                966
6       Dumbledoor          963
7       Travis              962
8       Travis              958
9       JarrodYo            955
10      Alex                950
```

You'll also notice the '?' button in the top right corner of the form; each of my forms has one of these, and when clicked, it will provide appropriate instructions on what the form does and how to use it. Here is the message box that is displayed when the '?' button is clicked on the Hall of Fame form, for example:



11SDD Major Project Mirrington

The buttons at the top of the screen sort the highscores according to difficulty level or the current player.
Press 'Normal', 'Epic' or 'Legendary' to view the Top 10 scores for that difficulty level.
Press 'My Scores' to view all of the previous scores for the player you are logged in as.
Press 'Back To Main Menu' to return to the main menu.

Next week, I will be working on a formal test data table for my project, as well as importing screenshots into my portfolio.

MAINTAINING

POSSIBLE IMPROVEMENTS/UPDATES

Whilst I am overall quite happy with how the project turned out, there are always more ways to improve and update the project; such is the nature of the software development process. Possible improvements/updates for the future may include:

- Add a timing feature that tracks the number of seconds taken to play each game. Use this to calculate scores or possibly display in hall of fame.
- Add a timed mode, where time is limited instead of the number of guesses.
- Add ability to search the hall of fame for any player, not just show the results for the currently playing player.
- Add in more difficulty levels, with differing numbers of guesses and a wider range of numbers.

Of course, if I decided that I wanted to implement any of these features, it would be necessary for me to follow through with the software development process again for each update; I would need to re-evaluate the new specifications of my project, design and plan the improvements, then implement, and test them again (and then possibly continue again for even more updates). In its current form, however, my project satisfies all original user requirements, and is thus ‘Complete’ (at least for now).