Name: Joseph Picchi

Project 3 Report

a) One notable obstacle I overcame was learning how to use assert statements in order to repeatedly check conditions in my code. At the start of the project, I was instead using my main function to input test strings on an individual basis, so I would have to keep inputting many test strings with each segment of additional code I wrote within isValidRowerString(). In that way, learning to use assert statements helped me to overcome a large hurdle in my project.

Another obstacle was staying organized with all of the many conditions in my isValidRowerString() function, especially when it came to detecting the source of various errors in my code. However, I eventually discovered that placing comments before large code blocks offered a great way for me to go back through my code and detect the areas that needed change.

A final obstacle was considering the vast number of potential test cases that could cause my program to produce erroneous results. The solution to this was primarily diligence and organization, as well as patience. Only by working on the program over the course of several days was I able to think of and test for each and every test case.

b)

The design of the program begins with the implementation of the “improved numbers function” provided on the ccle webpage. This was defined as the first function in my program, which I called “stringToNumber”.

The isValidRowerString() function was implemented next, and the general premise was that it takes in a string and checks a multitude of conditions that would cause the string to be false. If the string passes all of these conditions, then the function returns true. Its layout is as follows:

Return false if rowerString is empty.

Skip white space.

Make sure the first character is character between ‘1’ and ‘9’, or a colon. Otherwise, return false.

Record the minute value using the stringToNumber() function, and make sure the value satisfies the conditions in the spec.

Make sure there is a colon after the minutes value, followed by a number.

Make sure the seconds field is only two characters long.

Record the seconds value using stringToNumber(), and make sure the value is within the limits of the spec.

If seconds value is less than 10, make sure there is a 0 preceding the one digit number, so the format of the seconds field is of the form “:0x” where x is between 0 and 9.

Make sure there’s a white space after the seconds field.

Call stringToNumber() to record the strokes per minute, and make sure that value satisfies the limits of the spec.

Make sure the sequence of characters “ s/m” follows the stroke rate number.

Check for a white space after “ s/m”.

Call stringToNumber() to record the distance number, and make sure it is valid.

Make sure the sequence of characters “ m” follows the distance number.

Check for a white space after “ m”

Call stringToNumber() to record the heartrate, and make sure its value is valid.

Check for extra characters after the heartrate number.

If the string satisfies all of these properties, isValidRowerString returns true.

Next I created the totalDistance() function with the following pseudocode:

If isValidRowerString returns false, then return -1.

Find the position of the first m in the string, and add 1 to that position.

Call stringToNumber() to record the distance number.

Return that number.

The function heartRate() was designed as follows:

If isValidRowerString returns false, then return -1.

Find the position of the second m in the string, and add 1 to that position.

Call stringToNumber() to record the heartRate number.

Return that number.

The function strokesPerMinute() was designed as follows:

If isValidRowerString returns false, then return -1.

Find the location of the colon in the string, and add 3 to that index.

Call stringToNumber() to record the stroke rate number.

Return that number.

The elapsedMinutes function was designed as follows:

If isValidRowerString returns false, then return -1.

Call the stringToNumber() function to record the minutes value.

Return that number.

Lastly, the elapsedSeconds function was designed as follows:

If isValidRowerString returns false, then return -1.

Find the location of the colon in the string, and add 1 to that index.

Call the stringToNumber() function to record the seconds value.

Return that value.

c) A list of test data that could be used to thoroughly test my program (along with the reason for each test) is enumerated as follows:

|  |  |
| --- | --- |
| **List of Test Data** | **Reason for Each Test** |
| :14 28 s/m 42 m 110 | Each value is comfortably within the limits of what is considered valid. |
| :14 28 s/m 42 m 110 | Extra spaces at the start and between fields should be allowed. |
| 1:14 28 s/m 42 m 110 | Makes sure the program accepts the lower limit of the minute value (1) when minutes are present. |
| 59:14 28 s/m 42 m 110 | Makes sure the program accepts the upper limit of the minute value (59). |
| 59:00 28 s/m 42 m 110 | Makes sure the program accepts the lower limit of the seconds value (00). |
| 59:59 28 s/m 42 m 110 | Makes sure the program accepts the largest valid seconds value (59). |
| :04 28 s/m 42 m 110 | Makes sure the seconds are accepted in the proper format when the seconds value is less than 10. |
| 59:59 999 s/m 42 m 110 | Makes sure the program accepts the largest valid stroke value (999). |
| 59:59 1 s/m 42 m 110 | Makes sure the program accepts the smallest valid stroke value (1). |
| 59:59 999 s/m 42 m 1 | Makes sure the program accepts the smallest valid heart rate (1). |
| 59:59 999 s/m 42 m 999 | Makes sure the program accepts the largest valid heart rate (999). |
| asdf:14 28 s/m 42 m 110 | Makes sure the program rejects a string with a leading character other than ‘:’ or a digit. |
| :14 28 s/m 42 m 110 asdf | Makes sure the program rejects a string with extra ending characters. |
| :14 28 s/m 42 m 110 | The program must reject extra spaces after the stroke value. |
| 0:14 28 s/m 42 m 110 | The program must reject a minute value smaller than 1. |
| 0000:14 00028 s/m 00042 m 000110 | The program must reject leading extra zeros in any field. |
| :-14 -28 s/m -42 m -110 | The program must reject negative values in any field |
| 555:14 28 s/m 42 m 110 | The program should reject a minutes value larger than the maximum. |
| :555 28 s/m 42 m 110 | The program should reject a seconds value larger than the maximum. |
| :004 28 s/m 42 m 110 | The program should reject a seconds value less than ten with extra leading zeros. |
| :14 99123 s/m 42 m 110 | The program should reject a stoke rate higher than the maximum. |
| :14 0 s/m 42 m 110 | The program should reject a stroke rate lower than the minimum. |
| :14 28 s/m 42 m 99123 | The program should reject a heart rate higher than the maximum. |
| :14 28 s/m 42 m 0 | The program should reject a heart rate lower than the minimum. |
| :14 28 42 m 10 | The program should a reject a string without “s/m” after the stroke rate. |
| :14 28 s/m 42 10 | The program should reject a string without ‘m’ after the distance value. |
| :14 28 s/m 42 m | The program should reject a string that lacks any one value. |
| :4 28 s/m 42 m 234 | The program should reject a seconds field less than 10 that doesn’t contain a leading zero (format should be :04). |
| :4 28 s/m 0 m 234 | The program should reject a string with a distance value lower than 1. |
| 9:030 23 s/m 21 m 43 | The program should reject a string with a seconds field longer than 2 digits. |

The program I created handles all of this test data properly.

NOTE: When I stated “the program should reject” a given string, this means that the function isValidRowerString() should return false and all of the other functions should return -1 for the given string.