Com Sci 31 – Winter 2019

Project 3 Report

Name – Suyash Kumar

UID – 605108040

Discussion – 1C

# Major obstacles overcame

* Figuring out where to begin, since the project was such an involved one.
* Understanding how the number() function works and implementing it to create other functions, most importantly the isValidRowerString() function.
* Creating the isValidRowerString() function, which constituted the backbone of the whole program. The isValidRowerString() function seeks to confirm whether a string is good or not step by step – first it tries to check for minutes, then seconds, then strokes per minute, then distance, and then finally heartrate. If any of these fields are found to be improperly formed, the isValidRowerString() function returns false immediately.
* Using the number function and isValidRowerString() functions together for creating other functions. The functions created – elapsedMinutes(), elapsedSeconds(), strokesPerMinute(), totalDistance(), and heartRate() – were in increasing order of complexity.
* Undertaking the program development in phases instead of going out all at once. Regularly debugging and checking if the program developed so far is working well or not.
* Extensively testing the program to ensure its robustness against erroneous test cases as well as fair handling of good test cases.

# Program description

number() function

int number (parameter string, updatable position)

{

while(leading spaces remain)

{

update position by one

}

if(digit between 1 and 9)

{

update the return quantity to this digit by converting to int

while(next character is a digit between 0 and 9)

{

keep updating the return quantity to include this digit and updating position as well

}

}

else

{

return error value of -1

}

}

isValidRowerString() function

bool isValidRowerString (parameter rowerString)

{

position = 0

number(rowerString), which may take the minutes value if correctly formed and update position

if(minutes between 0 and 59)

{

number(rowerString), which may take the seconds value if correctly formed and update position

if(seconds between 0 and 59)

{

number(rowerString), which may take strokesPerMinute value if correctly formed and update position

if(strokesPerMinute between 1 and 999)

{

number(rowerString), which may take distance value if correctly formed and update position

if(distance positive)

{

number(rowerString), which may take heartrate value it correctly formed and update position

if(heartrate between 1 and 999)

{

all fields are well formed, return true

}

else

{

heartrate field not well formed, return false

}

}

else

{

distance value erroneous, return false

}

}

else

{

strokesPerMinute is not well formed, return false

}

}

else

{

seconds is poorly formed, return false

}

}

else

{

minutes is badly formed, return false

}

}

elapsedMinutes() function

int elapsedMinutes(parameter rowerString)

{

if(isValidRowerString(rowerString))

{

minutes = number(rowerString)

return(minutes)

}

else

{

string not well formed, return -1

}

}

elapsedSeconds() function

int elapsedSeconds(parameter rowerString)

{

if(isValidRowerString(rowerString))

{

minutes = number(rowerString) and updates position

seconds = number(rowerString)

return(seconds)

}

else

{

String not well formed, return -1

}

}

strokesPerMinute() function

int strokesPerMinute(parameter rowerString)

{

if(isValidRowerString(rowerString))

{

minutes = number(rowerString) and updates position

seconds = number(rowerString) and updates position

strokesPerMin = number(rowerString)

return(strokesPerMin)

}

else

{

String isn’t well formed, return -1

}

}

totalDistance() function

int totalDistance(parameter rowerString)

{

if(isValidRowerString(rowerString))

{

minutes = number(rowerString) and update position

seconds = number(rowerString) and update position

strokesPerMin = number(rowerString) and update position

distance = number(rowerString)

return(distance)

}

else

{

String not well formed, return -1

}

}

heartRate() function

int heartRate(parameter rowerString)

{

if(isValidRowerString(rowerString))

{

minutes = number(rowerString) and update position

seconds = number(rowerString) and update position

strokesPerMin = number(rowerString) and update position

distance = number(rowerString) and update position

heartRate = number(rowerString)

return(heartRate)

}

else

{

String is not well formed, return -1

}  
}

# Test data

Good test cases

It makes sense to first check extreme cases:

1. Minimum value of all fields - :00 1 s/m 1 m 1
2. Maximum value of all fields (distance has no upper limit, although it cannot take values larger INT\_MAX) – 59:59 999 s/m 99999 m 999

We now test a couple of intermediary cases with arbitrary amount of leading or intermediate spaces

1. :08 22 s/m 29 m 119 (0 minutes and seconds between 0 and 9)
2. 00:21 390 s/m 99 m 120 (0 minutes and seconds between 10 and 59)
3. 4:00 32 s/m 32 m 870 (Integer number of minutes and zero seconds)

Bad test cases

It is essential to thoroughly test the program against several possible erroneous test cases

Formatting type errors

1. (Empty String)
2. 00:04 28 s/m 42 m 110 (Leading zeros in minutes field)
3. 49;52 89 s/m 32 m 989 (Using ‘;’ instead of ‘:’)
4. 9:17 290 s / m 201 m 123 (invalid “s / m” string)
5. 12:42 241 s/m 21m 424 (no space between distance value and ‘m’ character)
6. 13:41 299 s/m 12 m 300hi (characters after heartrate)
7. 14:4a 103 s/m 32 m 321 (letter character in seconds field)

Range errors

1. 1091:59 100 s/m 13 m 101 (out of bound minute value)
2. 11:109 190 s/m 32 m 111 (out of bound second value)
3. 15:42 -100 s/m 102 m 209 (out of range strokesPerMinute value)
4. 13:51 101 s/m 92 m 10919 (out of range heartRate value)
5. 91:108 1x9s/m 21am 1090 (multiple errors – out of range minute/ second/ heartrate values, and invalid strokesPerMinute and distance expressions)