Report

Jingjing Nie (304567417)

1. Some of the notable obstacles I overcame include checking strings by dividing the strings piece by piece and making the process suitable for a loop so that the string can always be checked no matter how long the input is. Also, how to deal with different cases in which some of them are valid and others are not is a challenge for me. Moreover, how to use loops to let the program check the long strings and when to break the loops are also challenges I overcame. Another big obstacles is to find out what goes wrong when the program cannot build or produces outputs that are inconsistent from the requirements.
2. **bool isWellFormedAirportString(string commands)**

The basic idea is to check the long string part by part. For instance, check the first two letters first, if the first two characters are valid, check the following digit numbers, and if the digit numbers are valid, check the following ‘+’’-’ sign and minute digits. Then, using loop, the program will do the same thing for the following airport string in the string input.

To check the first two characters, I clipped the first two characters off the input string to better compare them with the valid airline codes. Since both upper case and lower case are valid, I brought them all to lower case before making comparisons.

I introduced an integer i to keep track of the position of the string that are checked to be valid. It is initialized to be 0. After a portion of the string is tested and clipped, i is again initialized to be 0.

To check the following digits, I first use a loop to find the number of digit of flight number. Since the valid flight number can only have 1-3 digits, i can only be 1-3 to be valid. Then, since it cannot be zero, the sum of the digits has to be greater than one, two, or three ‘0’s.

Then, if the following character is ‘+’ or ‘-’, it is valid and i increases by one. After checking the flight number and sign, I clipped the part off the string and set i to be 0.

Further arrival time digits are checked by first identifying the number of digits present. Similarly, if the character is a digit, i increments by one, and since the minute can be 0-999, i can only be 1-3.

If the above parts of the string are valid, the program has checked the first subunit of the input string. Therefore, this valid part should be cut off from the input string and a loop and the remaining parts are checked in the same way with the while loop.

The while loop follows the idea that the loop breaks if one part of the string is checked to be invalid, and returns false. On the other hand, if the string is valid, the loop breaks when all the characters are checked (the cut remaining part is empty) and returns true.

**double ontimeArrivalPercentage(string commands, string airlinecode)**

The basic logics involved in this function is similar to the above one, that is to check parts by parts

Similarly, since both upper and lower cases are valid, I converted both “commands” and “airlinecode” to lower case letters. Then, check for the validity of “airlinecode” input by comparing it to the accepted ones and check for the validity of “commands” by calling the previous function. If any of these two strings are invalid, I set the “OntimeArrivalPercentage” to be -1.

Then, check for the airline code by clipping off the first two characters of airport string. If they do not match, clip the string associated with this code off the string by identifying the length, and continue the loop.

If they match, “total”---total airlines with this code increases by one. Then, we check for the sign of the arrival time. If the flight number with ‘-’ minute is considered as on time, so “Ontime” --- flight with arrival time within 15 minutes, increases by 1, and the sign is clipped off the string for further check.

If the sign is ‘+’, then we check for the time digit after clipping off the ‘+’ sign. The arrival time can be 1, 2, or 3 digits. Thus, switch between minute:

Case 1: (1 digit) is already smaller than 15mins, so “Ontime” increases by 1.

Case 2: if the first digit is 0, “Ontime” +1, or if the first digit is 1 and the next digit is 0-4, “Ontime” +1.

Case 3: if the first is 0 and the next two follow case 2, or if both the first and second digit are 0, “Ontime” +1.

After this part is checked, cut it off the string and the loop continues until the remaining string is empty.

OntimeArrivalPercentage = Ontime / total

If “total” is 0, set the “OntimeArrivalPercentage” to be 0.

1. Test cases:

For function “**bool isWellFormedAirportString(string commands)**”

“DA132+90” Should return false because airline code is invalid.

“AA1345+0” Should return false because fight number exceeds 999.

“vA00-0” Should return false because flight number cannot be 0 or 00 or 000.

“Ua798+10000” Should return false because the arrival time exceeds 999.

“aa456\_78” Should return false because ‘+’ or ‘-’ character is missing.

“Aa135+309SW235712” Should return false because the second flight number is invalid

“DL900+0Ua123+780Aa123+092aa45+13” Should return true.

For Function “**double ontimeArrivalPercentage(string commands, string airlinecode)**”

“DA132+90”for airport string. Should return -1 because airport string is invalid, and so all cases that return false in the above function should return -1 for this function.

“Aa153+8” for airport string and “AB” for airline code. Should return -1 because the airline code is invalid.

“Aa153+8” for airport string and “AL” for airline code.” Should return 0 because the airport string does not contain AL airline, so no match is found.

“DL900+0Ua123+780Aa123+092aa45+13” for airport string and “aa” for airline code. Should return 0.5 because there are 2 aa airlines in the airport string and 1 of 2 is on time.

“aL900+90Ua123+780Aa123+092aa45+13Al290-89” for airport string and “al” for airline code. Should return 0 because there are 2 aa airlines in the airport string and 0 of 2 is on time.