CS 602
Algorithm Design and Implementation

Assignment 3

[Question 1] Longest Common Mountain Subsequence

Given two sequences $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and $B = \{b_1, b_2, \ldots, b_{|B|}\}$, a sequence $C = \{c_1, c_2, \ldots, c_{|C|}\}$ is called a common mountain subsequence of A and B, if (i) there exist two strictly increasing functions f and g (meaning $f(x_1) < f(x_2)$ if $x_1 < x_2$), such that $c_i = a_{f(i)} = b_{g(i)}$ for all $1 \le i \le |C|$; and (ii) there exists an index k, such that $c_1 < c_2 < \ldots < c_k > c_{k+1} > \ldots > c_{|C|}$, for some k in $1 \le k \le |C|$. There are many such common mountain subsequences between A and B, however, there is a longest subsequence among them. In this question, you are to write a program which computes the length of a longest common mountain subsequence.

Implement an algorithm in the function `LCMS`. Please also state and justify the time complexity of your algorithm as comments in your code.

Test inputs begin with the number of pairs of sequences. Each sequence is described by one line and contains a list of positive integers in hexadecimal between 1 and $2^{16}-1$. The length of each sequence is at least 1 and at most 1000. For each pair of sequences, output the length of a longest common mountain subsequence.

Sample Input

```
3
67cd 8c8e cc86 3fca f1c6 1dcb a6e1 7887 29b1
7c89 5a30 3fca db49 7ffb 7afc 1dcb 8ab3 29b1
67 fee3 9efa aa01 224 feb3 6279 ed2d 3db2 692 1fc5 75d
a7a6 9bac 67 a47e fee3 feb3 c4d 692 ddab 1fc5 4557
379a 5904 7589 c19d a5bb 81a7 9fee 736b 3c5d 4865 138 545 3bb 260d 35c
5904 c19d a0c5 9fee b6b 3c5d 8abb 5941 4865 5795 c311 545 260d e8c0 b09
```

Sample Output

```
2
4
5
```

[Question 2] Petrol or Battery?

Hybrid cars are getting more and more popular in the city of Temapura, which can either run on petrol ("petrol mode") or run on battery ("battery mode") when battery is enough. Driver can switch between petrol mode (cars run on petrol) and battery (cars run on battery) mode several times during one trip. However, if the battery is not enough, the car can only travel on petrol mode. In addition, the battery can be charged when the car is on petrol mode. Your task is to decide the most economical energy option for a trip by switching between petrol mode and battery mode. To simplify the problem, let's make these assumptions:

- There is only one route for each trip.
- There are pedestrian crossings along the road, and cars always need to stop before the crossings. Other than these crossing, cars do not stop, there is no traffic congestions either.
- Cars can only switch mode when it is stopped. Every switch costs 1% of the battery. It applies to the start of the trip as well: no matter what mode is chosen for the first segment, 1% of the battery is spent on starting the motor.
- Cars will not switch to battery mode if the battery level is less than 10% after a switch, and must switch to petrol if its battery is below 10% before a pedestrian crossing.
- On flat road, for every 10km cars travelled on petrol mode, the battery can be charged for 1%. This conversion rate can drop to as low as 0.8% on up slopes or increase to as high as 1.2% on down slopes.
- Cars need 1 liter of petrol to travel 10km, and 1 liter of petrol costs 2 dollars. In contrast, in battery mode, cars can travel 5km on 1% of battery, and 1% of battery only costs 1 dollar. The total cost of the journey is calculated the dollar amount spend on petrol plus electricity as you need to charge the car to its initial battery level. For example, if the battery level is at 85% at the start of the trip, and dropped to 75% when you finish the trip, you need to pay 10 dollars for 10% of battery. However, if the battery is not less than 85% at the end of the trip, there is no charge on electricity.

Design a dynamic programming algorithm for this task. Test inputs begin with a number indicating the number of lines below. Each line begins with a number representing the initial battery level and an integer representing the number of road segments *s* (*s-1* crossing divides the road into *s* segments, $s \leq 100$). This two numbers are then followed by pairs of numbers, in the format of "d_1:r_1 d_2:r_2 ... d_s:r_s", where d_i is the distance of road segment i ($d_i \leq 40$), and r_i is the ratio of the conversion rate depending on the slope of road segment i. For computational simplicity, r_i is only related to charging batteries, but not related to the using batteries. For each line of input, please output the minimum amount in dollar values for the trip, rounded to 2 decimal places. Take the first row of the sample input for example, battery mode for the first two segments and petrol mode for the last segment gives minimum charge of 6.60.

Sample Input

```
3
85 3 3:0.88 5:1.09 30:1
98 6 14:0.94 27:0.8 20:0.96 22:0.84 10:1.14 20:1.08
55 10 17:0.84 24:1.14 13:1.09 19:0.93 13:1.16 15:1.2 2:0.85 4:0.84 12:0.86 18:0.84
```

Sample Output

```
6.60
17.22
19.86
```

Grading Rubrics:

1. There are 10 marks for each question.
2. There are 8 test cases for question 1, where the first test case is the sample input A3Q1.in. You get 1 mark for each test case if your code does not produce errors including wrong answer or time limit exceeded. The remaining 2 marks are assigned to the time complexity part, you should state the time complexity of your own algorithm, not an algorithm that you think might work, but not implemented.
3. There are 10 test cases for question 2, where the first test case is the sample input A3Q2.in. You get 1 mark for each test case if your code does not produce errors including wrong answer or time limit exceeded.

Running python skeleton with sample input:

1. Open "Anaconda Prompt"
2. Go to the directory where you put the file **A3Q1.py** and **A3Q1.in**, using command **`cd`**
3. Run command **python A3Q1.py < A3Q1.in**
4. You may want to create a test input called **my_own_test.in** to design a test case for your own program, the command would then be **python A3Q1.py < my_own_test.in**
5. Same applies to Question 2, so you may run **python A3Q2.py < A3Q2.in**
6. Make sure your program read input from **`sys.stdin`**. If you find it difficult to work with **`sys.stdin`**, you might write the function that your need to modify with your own input and output, and then copy and paste to the function itself to the code skeleton provided. Please test with the command in item 3 to ensure it produces the right output.