**C4M Homework: Week 2, Part 2: Loops**

For some of these problems, you will submit your solutions on MarkUs `https://markus.cdf.toronto.edu/c4m-2016` as you did for the homework for workshop 1. For others you will submit on PCRS, `https://teach.cdf.toronto.edu/c4m` as you did for part 1 of the homework for workshop 2.

1. A list represents the queue of patients who signed up to see a doctor at a walk-in clinic. A patient would sometimes sign up twice in a row by mistake. Write a function with the signature `two_in_a_row(queue)` that returns `True` if a list contains the same name twice in a row, and `False` otherwise. For example, `two_in_a_row(["Mike G.", "Alice C.", "Bob A.", "Bob A.", "Mary C."])` should return `True`, since the string `"Bob A."` appears twice in a row, but `two_in_a_row(["Sam A.", "Dorothy Z."])` should return `False`.

   Submit the function on PCRS in the Workshop 2 Homework 2 quest.

2. A patient's blood pressure is measured at every check-up, and is recorded as `"low"`, `"high"`, or `"normal"`. The date of each check-up is also recorded. We are interested in knowing the date on which a sequence of 3 or more `"high"` measurements has started,

   Write a function with the signature `onset_high_blood_pressure(bp, dates)` which returns the date of the first time that a sequence of three or more `"high"` measurements was observed. If there is no such sequence, return the string `"No high blood pressure period observed"`.

   For example, if

   ```
   bp =    ["low",  "low",  "high", "normal", "high",  "high", "high", "high", "normal"]
   dates = ["12/3", "20/3", "29/3", "5/4",    "15/4",  "30/4", "10/5", "17/5", "29/5"]
   ```

   `onset_high_blood_pressure(bp, dates)` should return `15/4`.

   Write your function in Pyzo. Test it with the example above. Then write some other test cases for your function. One rule that is sometimes used for testing, is that your full set of tests must at least execute every line of code in your function. That way no line goes completely untested. Make sure that your tests do this at a minimum. Put your function in a file `bp.py` and submit it on MarkUs.

3. A patient's core body temperature is recorded at regular intervals, and stored as a list of `float`s. Write a function with the signature `consistent_growth_at_least_5(temps)` that returns `True` iff there was a period of *at least* 5 hours during which the temperature increased every hour.

   Submit the solution to this problem on PCRS where we have provided some tests. We recommend that you design your solution and do your own initial testing on Pyzo before working in PCRS.

4. Write a function with the signature `starts_with(s1, s2)` that takes in two strings `s1` and `s2`, and returns `True` if `s1` starts with `s2` and `False` otherwise.

   Submit this function on MarkUs in a file named `prefix.py`.

5. Ultimately this problem is to write a function with the signature `match_dna(seq1, seq1)` that takes in two dna sequences (as strings), and returns `True` if ond of them is a subsequence of the other (i.e., if `seq1` is a subsequences of `seq2`, or vice versa). In order to write your solution, you must use a function `match_at` at *every* possible index (using a for-loop).

   This problem is on PCRS. First we ask you to code `match_at` and then in the next question we provide our solution to `match_at` in the starter code for `match_dna`. In your solution to `match_dna`, don't change `match_at`, just use it.

   We have also posted 2 videos that together walk through the development of our solution to `match_at`. If you have trouble with your own `match_at`, you are encouraged to go ahead and watch the videos even if your solution doesn't pass all the test cases. You'll get more out of the video though if you at least attempt `match_at`.

6. A patient's core body temperature is recorded at regular intervals, and stored as a list of `float`s. Write a function with the signature `consistent_growth_exact_n(temps, n)` that returns `True` iff there was a period of *exactly* n hours during which the temperature increased every hour. (I.e., if the temperature increased n+1 periods, it doesn't count.)

   Write your code on Pyzo and include the function in a file called `growth.py`. In addition to the function definition, the file should have at least 3 calls to the function on different test input values. Submit your file on MarkUs.