X

Week 2 Programming Assignment

Due on 2020-02-15, 23:59 IST

Write three Python functions as specified below. Paste the text for all three functions together into the submission window. Your function will be called automatically with various inputs and should return values as specified. Do not write commands to read any input or print any output.

- You may define additional auxiliary functions as needed.
- In all cases you may assume that the value passed to the function is of the expected type, so your function does not have to check for malformed inputs.
- For each function, there are normally some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases. There are 12 private test cases, with equal weightage. You will get feedback about which private test cases pass or fail, though you cannot see the actual test cases.
- Ignore warnings about "Presentation errors".

---

1. A positive integer m can be expresseed as the sum of three squares if it is of the form p + q + r where p, q, r ≥ 0, and p, q, r are all perfect squares. For instance, 2 can be written as 0+1+1 but 7 cannot be expressed as the sum of three squares. The first numbers that cannot be expressed as the sum of three squares are 7, 15, 23, 28, 31, 39, 47, 55, 60, 63, 71, … (see Legendre's three-square theorem

Legendre's three-square theorem (https://en.wikipedia.org/wiki/Legendre%27s_three-square_theorem)).

Write a Python function `threesquares(m)` that takes an integer `m` as input and returns `True` if `m` can be expressed as the sum of three squares and `False` otherwise. (If `m` is not positive, your function should return `False`.)

Here are some examples of how your function should work.

```
>>> threesquares(6)
True

>>> threesquares(188)
False

>>> threesquares(1000)
True
```

2. Write a function `repfree(s)` that takes as input a string `s` and checks whether any character appears more than once. The function should return `True` if there are no repetitions and `False` otherwise.

   Here are some examples to show how your function should work.

```
>>> repfree("zb%78")
True

>>> repfree("(7)(a")
False

>>> repfree("a)*(?")
True

>>> repfree("abracadabra")
False
```

3. A list of numbers is said to be a hill if it consists of an ascending sequence followed by a descending sequence, where each of the sequences is of length at least two. Similarly, a list of numbers is said to be a valley hill if it consists of an descending sequence followed by an ascending sequence. You can assume that consecutive numbers in the input sequence are always different from each other.

   Write a Python function `hillvalley(l)` that takes a list `l` of integers and returns `True` if it is a hill or a valley, and `False` otherwise.

   Here are some examples to show how your function should work.