22C:16 (CS:1210) Quiz 4

You have 20 minutes to complete this quiz.

1. Consider the following expressions. For each expression, first determine whether the expression is *well-formed* or not. For each well-formed expression, determine if the expression will be successfully evaluated by Python or not. For each expression that is successfully evaluated, determine its value and the type of its value. Assume that the `math`, `random`, and `sys` modules have all been imported prior to the execution of these expressions. Also, suppose that the value of `sys.maxint` is `9223372036854775807`.

   (a) `len(str(random.randint(1, 4)*22))`

   (b) `sys.maxint+2`

   (c) `5 < 8 and (100/(len("0")-1))`

   (d) `abs(5 - 25) < 15 and 0.5`

   (e) `2/math.trunc(1.5)+200L`

   (f) `5 < 8 or (100/(len("0")-1))`

   (g) `bin(len(str(10<20)))`

   (h) `4L*len("Problem1")/(len("Exam1")%3)+2.0`

   (i) `-4*-3**2+-len(str(66))`

   (j) `bool(math.sqrt((-1)**100))`

2. We have defined a function called `factorSum` whose header is:

```
def factorSum(n):
```

The parameter `n` is expected to be a positive integer and the function returns the sum of all the factors of `n`. For example, if `n` were 10, the function would return 18 (which is 1 + 2 + 5 + 10).

Now, here is a definition. A positive integer n is called *perfect* if the sum of all the factors of n, excluding n, is equal to n. For example, $n = 6$ is perfect because its factors, excluding 6, are 1, 2, and 3 and 1+ 2 + 3 = 6.

For this problem, we want you to write a function called `nextPerfect` with the following function header:

```
def nextPerfect(M):
```

You can assume that the parameter `M` is a positive integer. This function should return the smallest perfect integer that is greater than or equal to `M`. This function should call `factorSum` repeatedly to complete its task.

Here is an example that will help you understand what is being asked of you.
Example. The first four perfect integers are 6, 28, 496, and 8128. So if we call `nextPerfect(10)`, it should return 28. Even if we call `nextPerfect(20)`, the function should return 28. In fact, even if we call `nextPerfect(28)`, the function should return 28. However, if we call `nextPerfect(29)`, the function should return 496.