6.00 Introduction to Computer Science and Programming
Fall 2008

6.00: Introduction to Computer Science and Programming

Problem Set 2

Introduction

This problem set will introduce you to using control structures in Python and using exhaustive search as a problem solving technique.

Collaboration

You may work with other students. However, each student should write up and hand in his or her assignment separately. *Be sure to indicate with whom you have worked.* For further detail, please review the collaboration policy as stated in the syllabus.

McDiophantine: Selling McNuggets

In mathematics, a **Diophantine equation** (named for Diophantus of Alexandria, a third century Greek mathematician) is a polynomial equation where the variables can only take on integer values. Although you may not realize it, you have seen Diophantine equations before: one of the most famous Diophantine equations is:

$x^n + y^n = z^n$.

For n=2, there are infinitely many solutions (values for x, y and z) called the Pythagorean triples, e.g. $3^2 + 4^2 = 5^2$. For larger values of n, Fermat's famous "last theorem" states that there do not exist any positive integer solutions for x, y and z that satisfy this equation. For centuries, mathematicians have studied different Diophantine equations; besides Fermat's last theorem, some famous ones include Pell's equation, and the Erdos-Strauss conjecture. For more information on this intriguing branch of mathematics, you may find the Wikipedia article of interest.

We are not certain that McDonald's knows about Diophantine equations (actually we doubt that they do), but they use them! McDonald's sells Chicken McNuggets in packages of 6, 9 or 20 McNuggets. Thus, it is possible, for example, to buy exactly 15 McNuggets (with one package of 6 and a second package of 9), but it is not possible to buy exactly 16 nuggets, since no non-negative integer combination of 6's, 9's and 20's adds up to 16. To determine if it is possible to buy exactly *n* McNuggets, one has to solve a Diophantine equation: find non-negative integer values of *a, b,* and *c*, such that

$6a + 9b + 20c = n$.

Problem 1.

Show that it is possible to buy exactly 50, 51, 52, 53, 54, and 55 McNuggets, by finding solutions to the Diophantine equation. You can solve this in your head, using paper and pencil, or writing a program. However you chose to solve this problem, list the combinations of 6, 9 and 20 packs of McNuggets you need to buy in order to get each of the exact amounts.

Given that it is possible to buy sets of 50, 51, 52, 53, 54 or 55 McNuggets by combinations of 6, 9 and 20 packs, show that it is possible to buy 56, 57,..., 65 McNuggets. In other words, show how, given solutions for 50-55, one can derive solutions for 56-65.

Theorem: If it is possible to buy x, x+1,..., x+5 sets of McNuggets, for some x, then it is possible to buy any number of McNuggets >= x, given that McNuggets come in 6, 9 and 20 packs.

Problem 2.

Explain, in English, why this theorem is true.

Save your answers for problems 1 and 2 as `ps2.txt`.

Solving a Diophantine Equation

Using this theorem, we can write an exhaustive search to find the largest number of McNuggets that cannot be bought in exact quantity. The format of the search should probably follow this outline:

```
Hypothesize possible instances of numbers of McNuggets that cannot be purchased
   exactly, starting with 1
For each possible instance, called n,
      Test if there exists non-negative integers a, b, and c, such that
         6a+9b+20c = n. (This can be done by looking at all feasible
         combinations of a, b, and c)
      If not, n cannot be bought in exact quantity, save n
When you have found six consecutive values of n that in fact pass the test of
   having an exact solution, the last answer that was saved (not the last value
   of n that had a solution) is the correct answer, since you know by the
   theorem that any amount larger can also be bought in exact quantity
```

Problem 3.

Write an iterative program that finds the largest number of McNuggets that **cannot** be bought in exact quantity. Your program should print the answer in the following format (where the correct number is provided in place of <n>):

`“Largest number of McNuggets that cannot be bought in exact quantity: <n>”`

Hint: your program should follow the outline above.

Hint: think about what information you need to keep track of as you loop through possible ways of buying exactly n McNuggets. This will guide you in deciding what state variables you will need to utilize.

Save your code for Problem 3 in `ps2a.py`.

We can generalize this idea to work with any size packages of McNuggets, not just 6, 9, and 20. For simplicity, however, we will assume that McDonald’s still provides McNuggets in three different sized packages.

Problem4.

Assume that the variable packages is bound to a tuple of length 3, the values of which specify the sizes of the packages, ordered from smallest to largest. Write a program that uses exhaustive search to find the largest number (less than 200) of McNuggets that cannot be bought in exact quantity. We limit the number to be less than 200 (although this is an arbitrary choice) because in some cases there is no largest value that cannot be bought in exact quantity, and we don't want to search forever. Please use ps2b_template.py to structure your code. Have your code print out its result in the following format:

```
"Given package sizes <x>, <y>, and <z>, the largest number of McNuggets that
cannot be bought in exact quantity is: <n>"
```

Test your program on a variety of choices, by changing the value for packages. Include the case (6,9,20), as well as some other test cases of your own choosing.

Save your code for Problem 4 in `ps2b.py`.

Hand-In Procedure

1. Save

Save your written answer in `ps2.txt` and your code in `ps2a.py` and `ps2b.py`, as instructed above. *Do not ignore this step or save your file(s) with different names.*

2. Time and Collaboration Info

At the start of each file, in a comment, write down the number of hours (roughly) you spent on the problems in that part, and the names of the people you collaborated with. For example:

```
 # Problem Set 2 (Part I)
 # Name: Jane Lee
 # Collaborators: John Doe
 # Time: 1:30
 #
... your code goes here ...
```