CSC236 tutorial exercises, Week #6
best before Friday afternoon

Please attempt to solve these exercises *before* tutorial on Friday. During tutorial, your TA will answer questions about the problems, and review solutions to selected problems. These exercises are not graded, so you are encouraged to work with classmates on them, and discuss them on the course discussion board.

1. Recall the function `subset_sum` which we saw in lecture on week 4:

```
1 def subset_sum(A, target):
2   if len(A) == 0:
3     return target == 0
4   return subset_sum(A[1:], target) or subset_sum(A[1:], target-A[0])
```

   Informally, `subset_sum` is supposed to return a boolean corresponding to whether or not there exists a subset of A, a list of integers, which adds up to the given integer $target$. For example, `subset_sum([4, 2, -1], 3)` should return True, and `subset_sum([4, 2, -1], 0)` should return False.*

   > ***Edited (02/14)**: Whoops, `subset_sum([4, 2, -1], 0)` should actually return True (we consider the empty list to be a valid subset, which sums to zero). A better example of an invocation that should return False would be `subset_sum([4, 2, -1], 7)`.

   (a) Devise a precondition for `subset_sum`. (It should be no more restrictive than necessary.)

   (b) Consider the following proposed postcondition for `subset_sum`:

   Postcondition: `subset_sum(A, target)` returns True if there exists a set of indices, I, such that $\sum_{i \in I} A[i] =$ target.

   Explain why this postcondition does *not* adequately describe the intended behaviour of `subset_sum`. Hint: think about other algorithms that would also satisfy this postcondition.

   (c) Devise a better postcondition for `subset_sum`. Your postcondition should be precise and unambiguous. (The failed postcondition from part b may be useful as a starting point.)

   (d) Use simple induction to prove that `subset_sum` is correct with respect to the precondition and postcondition you defined above.

2. Prove that the following function terminates on all valid inputs.

```
def mystery(n):
  """Pre: n is a positive integer
  Post: ???
  """
  if n == 1:
    return 1
  elif n % 2 == 1:
    return 1 + mystery(n+1)
  else:
    return 1 + mystery(n//2)
```