6.009 Quiz 1 -- Spring 2017

This quiz is **open-book and closed-internet**. Feel free to use any physical materials you have brought, but you may not access resources online (except web.mit.edu/6.009). Proctors will be available to answer administrative questions and clarify specifications of coding problems, but they should not be relied on for coding help.

Each test case is worth one point for a total of 20 points.

You *must* submit your modified `quiz.py` via web.mit.edu/6.009 before the deadline in order to receive credit. This quiz assumes you have Python 3.5 (or a later version) installed on your machine.

The `resources` directory contains the **Python Language Reference** and the **Python Library Reference** in PDF form. Please **do not import any Python modules** to use as part of your solutions -- quizzes with `import` statements (or their equivalent!) will be given a grade of 0.

As in the labs, `test.py` can be used to test your code in `quiz.py`. Each of the four problems has five tests. Remember that you can run specific tests by listing the test numbers, like so

```
python test.py 1 3 5      # run tests #1, #3, #5
```

Your score will be computed from the number of tests you pass. If you pass all the tests for a problem, you'll receive full credit. If you pass 2 of the 5 tests, you'll receive 40% of the credit for the problem. Note that to pass some of the tests in the time allotted, your solution may have to be reasonably efficient.

Problem 1: `findTriple( ilist )` (tests 1-6)

Find a triple of integers x, y, z in the list `ilist` such that x + y = z. Return the tuple (x, y). If the triple does not exist, return None. Note that x, y and z should be different elements of the list. For instance, for ilist=[1,5,2] the answer is None because we cannot use ilist[0] + ilist[0] = ilist[2]. In contrast, for ilist=[1,1,2], the answer is (1,1) because we can use ilist[0] + ilist[1] = ilist[2]. Also, note that the list may have more than one triple, but you should only return one. Examples:

`findTriple([4,5,9])` should return `(4,5) or (5,4)`.

`findTriple([20,40,100,50,30,70])` should return `(30,70) or (70,30)` or `(20,50) or (50,20)` or `(20,30) or (30,20)`.

`findTriple([6,11,7,2,3])` should return `None`.

Note: Using a triply-nested for loop will likely time out on the server.

Problem 2: `maximumSubsequence( ilist, is_circular=False )` (tests 7-12)

Given a list of integers `ilist` and the boolean `is_circular`, return a tuple of the start and end indices of the maximum subsequence in the list where the maximum subsequence is defined as the maximum sum of continuous integers in the list. If `is_circular` is True, imagine the list is circular. That is, after the end index comes the start index.

Examples:

`maxSubsequence([1,-5,2,-1,3])` should return `(2,4)` because the maximum subsequence is `[2,-1,3] = 4`. Note that is_circular is False in the invocation.

`maxSubsequence([1,-2,-3,4,5,7,-6])` should return `(3,5)` because the maximum subsequence is `[4,5,7] = 16`.

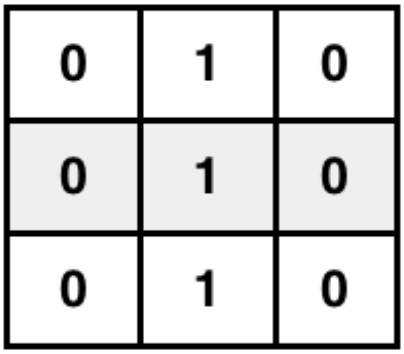
`maxSubsequence([4,-3,5,1,-8,3,1], True)` should return `(5,3)` because the maximum subsequence is `[3,1,4,-3,5,1] = 11`. Notice that the elements correspond to indices 5, 6, 0, 1, 2 and 3. For this example, if `is_circular` had been False, the answer would be `(0,3)` because the maximum subsequence that does not wrap around is `[4,-3,5,1] = 7`.

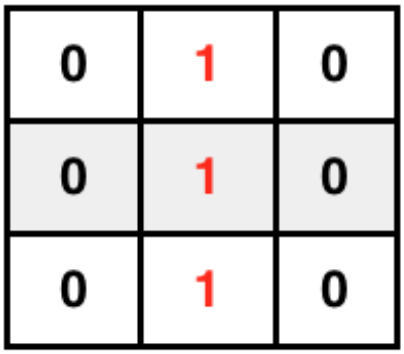
Problem 3: `findPath( grid )` (tests 13-20)

`grid` is a two dimensional m by n grid, with a 0 or a 1 in each cell. Find a path of ones starting at the top row (row 0) and ending at the bottom row (row n-1), where a valid move from cell `(r,c)` can go to: `(r+1,c-1)`, `(r+1,c)` or `(r+1,c+1)`. Return the path as a list of coordinate tuples (row, column). If there is no path, return None.

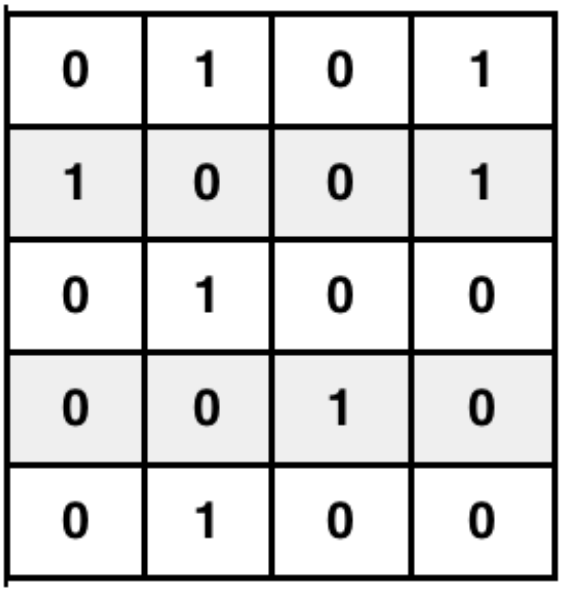
Examples:

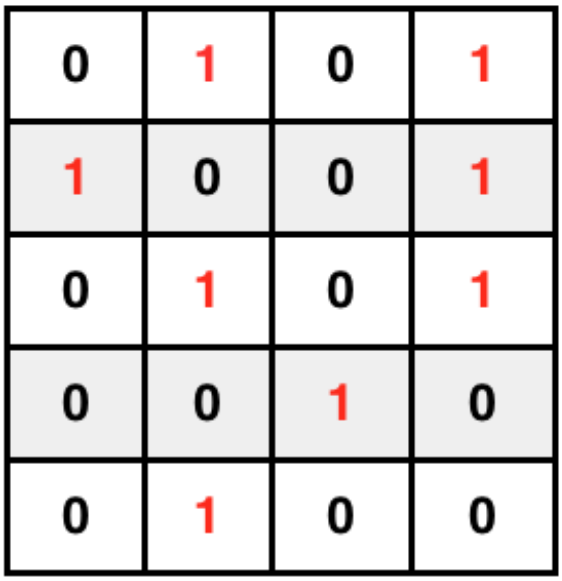
`findPath([[0,1,0],[0,1,0],[0,1,0]])` should return `[(0,1),(1,1),(2,1)]`.



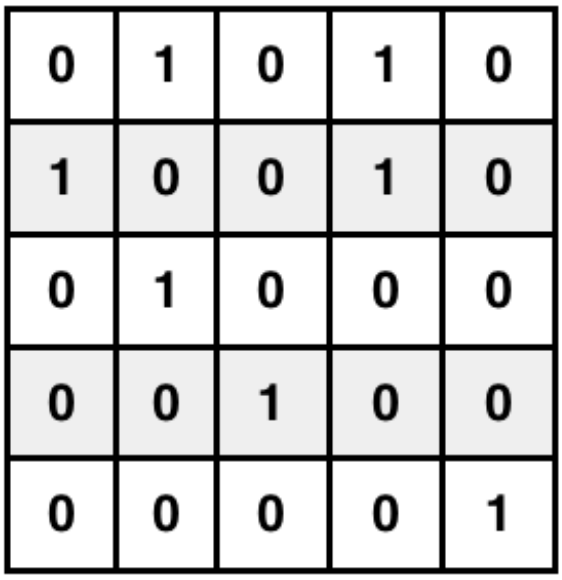


`findPath([[0,1,0,1],[1,0,0,1],[0,1,0,1],[0,0,1,0],[0,1,0,0]])` should return `[(0,1),(1,0),(2,1),(3,2),(4,1)]` or `[(0,3),(1,3),(2,3),(3,2),(4,1)]`.





`findPath([[0,1,0,1,0],[1,0,0,1,0],[0,1,0,0,0],[0,0,1,0,0],[0,0,0,0,1]])` should return `None`.



Note: In order to receive full credit, your code should be general enough to be used with arbitrary m and n, not just the test cases provided.