Note: for parts 2, 3 and 4 you have to submit your working code. We will test your code on a sample dataset.

1. Which kind of type equality does SML/NJ supports for tuples, records and lists? Show code examples to prove your answer? (10 points)

2. Write an SML program to find prime numbers using a tail recursion? Following are a few prime numbers that you can use for testing your program? (20 points)
Sample Primes = 38569, 79903, 98873, 74381.

3. Given a string of digits such as “12345”. Write a SML program that reverses the order of the digits e.g. “12345” changes to “54321” and so on? (20 points)

4. This question requires you to write SML code for manipulating representations of mobiles (picture placed below). We'll restrict ourselves to a very restricted set of mobiles with a binary branching structure.

At the top level, mobiles consist of two parts: a left branch and a right branch. Each branch also consists of two parts: a rod that has a length, and an object at the end of the rod. The object at the end of a rod can either be a simple weight or it can be a mobile.

Your task is to write a function called balanced that returns true if the single mobile that is passed to it as an argument is balanced and returns false otherwise. More precisely, when passed a representation of a single mobile it returns a “true” if that mobile is balanced and a “false” otherwise.
A mobile is balanced if and only if the torque on its top-left branch is equal to the torque on its top-right branch AND if the sub-mobiles hanging off of these top branches are themselves balanced. Assume that simple weights at the leaves are always balanced. The torque on a branch is equal to the weight at the end of the branch multiplied by the length of the branch. The weight of the branch is just the weight of the object hanging from it (assume the branch rods are themselves weightless).

(50 points)
Sample mobiles that you can use to test your code.

# The format for these mobiles is as follows:
#    [left_branch, right_branch]
#    where a branch is represented as
#    [rod_length, sub_mobile or leaf weight].
# Both rod_length and leaf weights are integers
#

good_1 = [[1,
            [[1,2], [2,1]],
            [3,1]]

good_2 = [[4,
            [[2,
              [[1,2], [1,2]],
              [2,
               [[9,1],
                [3,
                 [[5,2], [10,1]]]]],
              [8,
               [[1,
                 [[1,2], [2,1]],
                 [3,1]]]]
            ]]

bad_1 = [[5,3], [4,2]]

bad_2 = [[6, [[2,3], [4,2]]],
          [7, [[5,4], [7,5]]]]