Submission Instructions. Since this is a programming assignment, you will hand in your code electronically. To hand in your solutions, please copy your files to

/afs/cs.cmu.edu/academic/class/15499-s09/handin/username/assn2/

where username is your Andrew ID. Your function names should correspond to the function names given in the problem statement.

Problem 1: Scan

- Implement your own version of the Nesl pack function (call it myPack) using plus_scan with work complexity $O(n)$ and depth $O(\log n)$, or better. Assume that it only needs to work on vectors of integers. You can also assume that plus_scan has work $O(n)$ and depth $O(\log n)$.

  myPack([(13,F), (5,T), (34,F), (3,F), (8,T)])
  ⇒ it = [5,8] : [int]

- Develop an algorithm that given a string representing a mixture of text and numbers returns a t (true) wherever a character corresponds to a number and a f (false) otherwise. The string can contain letters ([a..z]), digits ([0..9]) or spaces. The trick is that digits that follow a letter are part of text and not a number. For example:

  parseNums("foo22 711")
  ⇒ it = [f, f, f, f, f, f, t, t, t] : [bool]

  since 771 belong to a number but 22 does not.

Problem 2: Stock Market

Given the price of a stock at each day for $n$ days, we want to determine the biggest profit we can make by buying one day and selling on a later day. For example:

  stock([12, 11, 10, 8, 5, 8, 9, 6, 7, 7, 10, 7, 4, 2])
  ⇒ it = 5 : int

since we can buy at 5 on day 5 and sell at 10 on day 11. This has a simple linear time serial solution. Write a Nesl program to solve the stock market problem with work complexity $O(n)$ and depth $O(\log n)$.

Problem 3: Magic Pointers

A pointer sequence is an integer sequence in which each position is interpreted as a pointer to another position in the sequence. For example, the sequence [1, 6, 2, 6, 2, 1, 6, 2] represents two trees with roots at 2 and 6. The sequence [1, 6, 5, 2, 0, 3, 4] represents two cycles (0 → 1 → 6 → 4 → 0 and 2 → 5 → 3 → 2).

Please write an algorithm in Nesl for the following problems. For parts a), b), and c), your algorithms should take $O(n \log n)$ work and $O(\log n)$ depth.

  a) Detect if a pointer sequence has any cycles (findCycle : [int] → bool)
b) For every node in a tree, return the position of the root of the tree \( \text{findRoots} : \ [\text{int}] \rightarrow \ [\text{int}] \) \\
c) Given a pointer sequence which only has cycles, return the number of cycles (\( \text{Hint: use the element with maximum index.} \)) \( \text{countCycles} : \ [\text{int}] \rightarrow \text{int} \) \\
d) Improve your algorithm in part c) so that it takes expected \( O(n) \) work and \( O(\log n) \) depth. \\
\( \text{countCyclesFast} : \ [\text{int}] \rightarrow \text{int} \)