Demonstration Task 3: Fruit

Jack and Jill are driving from Niagara Falls to Waterloo. Along the way, they pass several fruit stands. Each fruit stand sell baskets of cherries at one price, and baskets of peaches at another price. Jack wants to buy a basket of cherries, and Jill wants to buy a basket of peaches. They wish to stop at only one fruit stand. Which fruit stand should they stop at to minimize the total cost?

You are to implement a procedure **stopat(N,C,P)**. **N** is the number of fruit stands. The fruit stands are numbered from 0 to N-1. **C** and **P** are arrays representing the prices of cherries and peaches at each fruit stand, in dollars. **C[i]** is the price of a basket of cherries at the fruit stand numbered **i**. **P[i]** is the price of a basket peaches at the fruit stand numbered **i**. **stopat(N,C,P)** must return the number of the first fruit stand that has the lowest possible total price for a basket of cherries and a basket of peaches.

Subtask 1 [50 points]

Assume N≤10. The prices for baskets of cherries and peaches will be between 1 and 100 dollars each.

Subtask 2 [50 points]

Assume that N≤1000. The prices for baskets of cherries and peaches will be between 1 and 1000 dollars each.

Implementation Details

- Use the RunC programming and test environment
- Implementation folder: `/home/ioi2010-contestant/fruit/` (download prototype here)
- To be implemented by contestant: `fruit.c` or `fruit.cpp` or `fruit.pas`
- Contestant interface: `fruit.h` or `fruit.pas`
- Grader interface: *none*
- Sample grader: `grader.c` or `grader.cpp` or `grader.pas` *and* `graderlib.pas`
- Sample grader input: `grader.in.1` `grader.in.2`
  Note: The grader reads a sequence of pairs (C[0],P[0]), (C[1],P[1]), ... (C[N-1],P[N-1]), where N is the number of pairs.
- Expected output for sample grader input: `grader.expect.1` `grader.expect.2`
- Compile and run (command line): `runc grader.c` or `runc grader.cpp` or `runc grader.pas`
- Compile and run (gedit plugin): *Control-R*, while editing any implementation file.
- Submit (command line): `submit grader.c` or `submit grader.cpp` or `submit grader.pas`
- Submit (gedit plugin): *Control-J, while editing any implementation or grader file.*
- *CPU time limit: 10 seconds*
- *Memory limit: 256 MB*