Stock Predictions

Algorithms that can correctly predict stock prices can help generate millions of dollars. In this one-player game, you try to predict the rise and fall of the stock price of various stocks and buy or sell the stocks accordingly.

You start with $100. Each turn you will be given the stock prices of current day and previous 4 days. You must then choose to BUY or SELL the stocks. Your program will be run with input for 1 day at a time.

The stock prices are generated by us and may contain patterns.

Input Format

The input of each turn will consist of multiple lines. All money values are doubles to two decimal places, all other numbers are integers.

- The first line contains three space separated numbers m k d.
  - m - the amount of money you could spend that day.
  - k - the number of different stocks available for buying or selling.
  - d - the number of remaining days for trading stocks.
- k lines follow, each in the following format: name owned prices
  - name - the name of the stock (a string).
  - owned - the number of shares you own of that stock.
  - prices - 5 space separated numbers representing the stock's price for the last 5 days. These are ordered from oldest to newest, so the last number is the current stock price.

Your program will be fed the days sequentially so you can write to a file in order to store a longer history of the prices.

Output Format

The output for each turn should also contain multiple lines:

- Output N for the number of transactions you wish to make. Output 0 if you are not making any transactions that day.
- If you are making transactions, output N lines containing the name of the stock (case sensitive), BUY or SELL, and the number of shares you wish to buy or sell.

NOTE: Money earned from selling stocks will only become available (for buying stocks) on the following day.

Constraints

1<=k<=10

Sample Input

```
90 2 400
iStreet 10 4.54 5.53 6.56 5.54 7.60
HR 0 30.54 27.53 24.42 20.11 17.50
```

Sample Output

```
2
iStreet SELL 10
HR BUY 5
```

Explanation

You have 90 dollars and 10 iStreet stocks. You decide to sell the iStreet stocks, then purchase 5 HR stocks. After the transactions you will have 90 - 17.50 x 5 + 7.60 x 10 = 78.5 dollars and 5 HR stock for the next days trading.

You could not have bought more than 5 HR stock because you do not get to spend the money from selling iStreet stock until the next day.

Scoring

Your score is = to 5 x ln(money). You will only receive points from the hidden test case.

Task

You trade until you run out of days. Any stocks left on the last day will automatically be sold at the last day's market price. Your objective is to make as much money as possible starting from $100.

Complete the function printTransactions that takes a float *m* money left, an integer *k* number of stocks, an integer *d* days left, a string array *name* of stock names, an integer array *owned* of stocks owned, and an 2d integer array *prices* of stock prices containing *k* arrays of length 5 and print your output.

All indexes correspond properly (For index i, the name of the stock is *name[i]*, the number of shares of it you hold is *owned[i]* and the data about it is *prices[i]*.

Bigger data set

A bigger data set can be downloaded here. The data for the hidden test cases will be similiar.

Python 3

```python
from __future__ import division
from math import sqrt
from operator import add
from heapq import heappush, heappop

def printTransactions(money, k, d, name, owned, prices):
    def mean(nums):
        return sum(nums) / len(nums)

    def sd(nums):
        average = mean(nums)
        return sqrt(sum([(x - average) ** 2 for x in nums]) / len(nums))

    def info(price):
        cc, sigma, acc = 0, 0.0, 0
        for i in range(1, 5):
            if price[i] > price[i - 1]: cc += 1
        sigma = sd(price)
        mu = mean(price)
        c1, c2, c3 = mean(price[0:3]), mean(price[1:4]), mean(price[2:5])

        return (price[-1] - price[-2]) / price[-2]

    infos = map(info, prices)
    res = []

    drop = []

    for i in range(k):
        cur_info = info(prices[i])
        if cur_info > 0 and owned[i] > 0:
            res.append((name[i], 'SELL', str(owned[i])))
        elif cur_info < 0:
            heappush(drop, (cur_info, i, name[i]))
```

```
35
36      while money > 0.0 and drop:
37          rate, idx, n = heappop(drop)
38          amount = int(money / prices[idx][-1])
39          if amount  > 0:
40              res.append((n, 'BUY', str(amount)))
41              money -= amount * prices[idx][-1]
42
43      print (len(res))
44      for r in res:
45          print (' '.join(r))
46
47
48
49  if __name__ == '__main__':
50      m, k, d = [float(i) for i in input().strip().split()]
51      k = int(k)
52      d = int(d)
53      names = []
54      owned = []
55      prices = []
56      for data in range(k):
57          temp = input().strip().split()
58          names.append(temp[0])
59          owned.append(int(temp[1]))
60          prices.append([float(i) for i in temp[2:7]])
61
62      printTransactions(m, k, d, names, owned, prices)
```

Line: 62 Col: 53

Upload Code as File

Run Code | Submit Code

Game 1

Permalink

chetan_ai22

You finished with $10494.9. You earned 0 points because this is the sample case.

Download moves as file

Player: 1

Input	Output
7161.13 10 1 CAL 0 124.81 118.07 127.2 127.78 128.44 UCB 0 52.24 48.99 52.44 52.03 52.48 RIT 1 129.23 142.48 103.12 101.41 110.49 UCLA 0 9.01 27.8 27.78 28.22 32.43 USC 0 248.35 248.92 235.43 229.59 235.35 UFL 0 31 30.71 30.15 32.68 32.78 UMAD 0 130.15 149.47 105.53 142.96 105.68 RICE 0 121.86 127.12 117.78 125.38 124.22 UMD 0 97.83 104.11 95.66 98.36 103.2 UCSC 4 189.91 161.72 246.45 167.24 210.18	3 RIT SELL 1 UCSC SELL 4 UMAD BUY 67

Error