CSE 130 Exam
Stock Price Time-Travel Design

Pseudocode:

```
SET high_price <- 0                                                    A
SET time_high <- “”                                                    B
SET low_price <- 999999999999999999                                    C
SET time_low <- “”                                                     D

GET file_name from user
OPEN file_name as stock_price_file
  READ line as a full line of text from stock_price_file               E
    Split line into parts containing time and stock_price              F
    SET time <- parts[0]                                               G
    SET stock_price <- parts[1]                                        H

    // Condition to find highest stock price in the file along with the time it was high
    IF stock_price > high_price                                        I
      SET high_price <- stock_price                                    J
      SET time_high <- time                                            K

    // Condition to find lowest stock price in the file along with the time it was low
    IF stock_price < low_price                                         L
      SET low_price <- stock_price                                     M
      SET time_low <- time                                             N

  SET amount_shares_bought <- low_price * 100                          O
  SET amount_shares_sold <- high_price * 100                           P

  PUT In {time_low}, we found the lowest stock price. It was ${low_price}
  PUT In {time_high}, we found the highest stock price. It was ${high_price}
  PUT The best time to buy is at the lowest. Time travel to {time_low} to buy shares.
  PUT The best time to sell is at the highest. Time travel to {time_high} to sell shares.
  PUT Marty McFly bought 100 shares for ${amount_shares_bought} and made ${amount_shares_sold} when he sold 100 shares.

CLOSE file_name
END
```

Algorithmic Efficiency:
O(n) Efficiency/ Linear Performance
Linear algorithms are characterized by a loop where every element in the collection is visited once. Note that sometimes a library or a feature of the programming language may obscure this loop, but the loop must still exist.
Loop. There must be a loop controlled by the input in some way. Note that this loop could be hidden in recursion or it could be in a function that our code calls. • Every element visited. In most cases, linear algorithms visit every element in the input buffer. However, there are exceptions. An algorithm that visits every other element would still be O(n) but the equation would be cost = ½n.
I also say it is O(n) because the if statements under the loops are O(1). The complexity of the for loop (READ line) depends on the complexity of its component.

Program Trace:

Used the file, “stock-price.csv”. (Screenshot below)

stock-price.csv

Normal Arial

	A	B
1	Time	Stock Price $
2	1/1/01	234
3	2/1/01	89
4	3/1/01	21
5	7/1/02	123
6	8/1/05	60
7		

	high_price	time_high	low_price	time_low	time	stock_ price	amount _shares _bought	amount _shares _sold
A	0	/	/	/	/	/	/	/
B	0	0	/	/	/	/	/	/
C	0	0	999999999999999999	/	/	/	/	/
D	0	0	999999999999999999	0	/	/	/	/
E	0	0	999999999999999999	0	/	/	/	/
F	0	0	999999999999999999	0	/	/	/	/
G	0	0	999999999999999999	0	1/1/01	/	/	/
H	0	0	999999999999999999	0	1/1/01	234	/	/
I	0	0	999999999999999999	0	1/1/01	234	/	/
J	234	0	999999999999999999	0	1/1/01	234	/	/
K	234	1/1/01	999999999999999999	0	1/1/01	234	/	/
L	234	1/1/01	999999999999999999	0	1/1/01	234	/	/
M	234	1/1/01	234	0	1/1/01	234	/	/
N	234	1/1/01	234	1/1/01	1/1/01	234	/	/
E	234	1/1/01	234	1/1/01	1/1/01	234	/	/
F	234	1/1/01	234	1/1/01	1/1/01	234	/	/
G	234	1/1/01	234	1/1/01	2/1/01	234	/	/
H	234	1/1/01	234	1/1/01	2/1/01	89	/	/
I	234	1/1/01	234	1/1/01	2/1/01	89	/	/
L	234	1/1/01	234	1/1/01	2/1/01	89	/	/
M	234	1/1/01	89	1/1/01	2/1/01	89	/	/
N	234	1/1/01	89	2/1/01	2/1/01	89	/	/
E	234	1/1/01	89	2/1/01	2/1/01	89	/	/
F	234	1/1/01	89	2/1/01	2/1/01	89	/	/
G	234	1/1/01	89	2/1/01	3/1/01	89	/	/

H	234	1/1/01	89	2/1/01	3/1/01	21	/	/
I	234	1/1/01	89	2/1/01	3/1/01	21	/	/
L	234	1/1/01	89	2/1/01	3/1/01	21	/	/
M	234	1/1/01	21	2/1/01	3/1/01	21	/	/
N	234	1/1/01	21	3/1/01	3/1/01	21	/	/
E	234	1/1/01	21	3/1/01	3/1/01	21	/	/
F	234	1/1/01	21	3/1/01	3/1/01	21	/	/
G	234	1/1/01	21	3/1/01	7/1/02	21	/	/
H	234	1/1/01	21	3/1/01	7/1/02	123	/	/
I	234	1/1/01	21	3/1/01	7/1/02	123	/	/
L	234	1/1/01	21	3/1/01	7/1/02	123	/	/
E	234	1/1/01	21	3/1/01	7/1/02	123	/	/
F	234	1/1/01	21	3/1/01	7/1/02	123	/	/
G	234	1/1/01	21	3/1/01	8/1/05	123	/	/
H	234	1/1/01	21	3/1/01	8/1/05	60	/	/
I	234	1/1/01	21	3/1/01	8/1/05	60	/	/
L	234	1/1/01	21	3/1/01	8/1/05	60	/	/
E	234	1/1/01	21	3/1/01	8/1/05	60	/	/
O	234	1/1/01	21	3/1/01	8/1/05	60	2100	/
P	234	1/1/01	21	3/1/01	8/1/05	60	2100	23400

(In the trace table above, it’s highlighted in red whenever a variable change to a new value).