634947 2
For the selection sort algorithm; what is its best case and worst case running time?

a. Best: O(1)
Worst: O(n)

b. Best: O(n)
Worst: $O(n^2)$

c. Best: $O(\log_2 n)$
Worst: O(n)

*d. Best: $O(n^2)$

Worst: $O(n^2)$

e. None of the above.

f. "
g. "
h. "
i. "
j. "

General Feedback:

Selection sort repeatedly runs the Find-largest algorithm as its helper function. So, regardless of the list's initial ordering, Find-largest will cost n-1 comparisons for the first pass, n-2 for the second, etc. Hence selection sort's run time performence is independent of the list's initial ordering: $O(n^2)$

633400 2
You see the expression n = 100000 in some code that successfully compiles. What type can n not be?

a. int

*b. short

c. float

d. double

e. long

f. "
g. "
h. "
i. "
j. "

General Feedback:

Shorts can only hold values in [-32768, 32767].

633397 2
Suppose you try to perform a binary search on the unsorted array {1, 4, 3, 7, 15, 9, 24}. Which element will not be found when you try searching for it?

a. 7

b. 1

c. 9

*d. 15

e. 24

f. "
g. "
h. "
i. "
j. "

General Feedback:

The answer is 15. The first check will look at 7. 15 is greater than 7, so we search to its right. The second check will look at 9. 15 is greater than 9, so we search to its right. The third check will look at 24. 15 is less than 24, so we look to its left. However, our range has just been inverted and our searching stops, not having found 15.