617507
How many objects are created in the following declaration?

```
String[] names = new String[10];
```

a. 0

*b. 1

c. 10

d. 11

e. None of the above

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

General Feedback:

None of the above

632880
We say indexing is fast if it's done in O(1) time, searching is fast if done in O(lg N) time, and inserting and deleting are fast if done in O(1) time. Compared to other data structures, unsorted linked lists offer:

a. slow indexing, slow search, slow insertions and deletions.

b. fast indexing, fast search, slow insertions and deletions.

c. fast indexing, slow search, slow insertions and deletions.

*d. slow indexing, slow search, fast insertions and deletions.

e. slow indexing, fast search, fast insertions and deletions.

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

General Feedback:

Linked lists do not support fast indexing. To get at element i, one must traverse the list i steps. Slow indexing makes for slow searching, as the binary search relies upon fast indexing to retrieve the middle element. Insertions and deletions, on the other hand, can be done in constant time, as only one or two neighboring links are affected.

634930
Consider the following code snippet that copies elements (pixels) from one 2-d array (called pict), with maximum dimensions of maxWidth and maxHeight into another 2-d array called canvas.

```
canvas = [][]
for row in range (0, maxHeight):
    for col in range (0, maxWidth):
        origPixel = pict[col][row]
        canvas[width-1-col][row = origPixel
```

If one were to display canvas, how would it compare to the orignal picture (as represented by pict)?

*a. Rotated 180 degrees around the vertical center line/column.

b. Rotated 180 degrees around the horizontal center line/row.

c. Rotated left 90 degrees.

d. Rotated right 90 degrees.

e. Rotated 180 degrees.

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

General Feedback:

Each pixel remains on the same row (line) that it originally was on. For any given row, the leftmost pixel swaps positions with the rightmost. The second pixel in a row swaps positions with the second last pixel in that row, etc. Hence, the picture is rotated across the vertical center line.