2) (10 pts) DSN (Sorting)

The partition function in quick sort takes in an array, a low index, and a high index, which specifies a subsection of the array to partition, and returns the index where the partition element lies after performing the partition. Though there are many strategies to pick the partition element, to make grading easier, do the following: (a) use the element initially in index low to be the partition element, and (b) execute the in place partition where pairs of elements which are out of place are swapped and the partition element is swapped into its correct location at the very end right before the function returns this location. The swap function is provided for your use. **You may assume that low < high.**

```
void swap(int* ptrA, int* ptrB);

int partition(int array[], int low, int high) {

    int lowPtr = low+1, highPtr = high;

    while (lowPtr <= highPtr) {

        while (lowPtr <= high && array[lowPtr] <= array[low])
            lowPtr++;

        while (highPtr >= low && array[highPtr] > array[low])
            highPtr--;

        if (lowPtr < highPtr)
            swap(&array[lowPtr], &array[highPtr]);
    }

    swap(&array[low], &array[highPtr]);
    return highPtr;

}

void swap(int* ptrA, int* ptrB) {
    int temp = *ptrA;
    *ptrA = *ptrB;
    *ptrB = temp;
}
```

Grading: Code can be expressed in quite a few ways. Assign points to each of the following parts of the overall structure.

Outer loop set up with 2 indexes - 2 pts
Inner loop to advance low index - 2 pts
Inner loop to advance high index - 2 pts
Swap code for out of place elements - 2 pts
Last swap - 1 pt
Return - 1 pt