2) (10 pts) DSN (Sorting)

Consider the problem of sorting a competition struct. A competition struct has three integer components: **`probSolved`**, **`totalTime`** and **`difficulty`**. One struct is greater than another if has more problems solved (**`probSolved`**) than another. If the problems solved is the same between two structs and the total time is less, then that struct is greater than the other. Finally, between two structs with the same number of problems solved and total time, if one has greater difficulty, it is greater than the other struct. If all three components are equal, the structs are equal and neither is greater than the other. Write a function called greaterThan, which takes in pointers to two competition structs and returns 1 if the struct pointed to by ptrA is greater than the struct pointed to by ptrB, according to these rules, and 0 otherwise. For example, (pS=3, tT = 200, d = 8) is greater than (3, 200, 7) but is NOT greater than (3, 199, 7). Rather (3, 199, 7) is greater than (3, 200, 8).

```
typedef struct competition {
    int probSolved;
    int totalTime;
    int difficulty;
} competition;

int greaterThan(competition* ptrA, competition* ptrB) {

    // Grading: 1 pt if, 1 pt return.
    if (ptrA->probSolved > ptrB->probSolved) return 1;

    // Grading: 1 pt if, 1 pt return.
    if (ptrA->probSolved < ptrB->probSolved) return 0;

    // Grading: 1 pt if, 1 pt return.
    if (ptrA->totalTime  <  ptrB->totalTime) return 1;

    // Grading: 1 pt if, 1 pt return.
    if (ptrA->totalTime  >  ptrB->totalTime) return 0;

    // Grading: 2 pts total
    if (ptrA->difficulty > ptrB->difficulty) return 1;
    return 0;
}
```

Grading Note: The order of these statements matters (some), if each statement needed is present, but the order makes the code incorrect, then take off an appropriate number of points for the incorrect order (say 1, 2 or 3 pts depending on the severity.)