2) (10 pts) DSN (Sorting)

The critical part of the Quick Sort algorithm is the partition. One important part of the partition is the selection of the partition element. In general, it's better to have a "middle value" relative to the other values in the subarray to be sorted to be chosen as the partition element. One simple strategy to improve the probability of a good partition element is to select three items at random from the subarray to be sorted and use the middle value of those three as the partition element. In this particular problem, complete the function below so that it takes in an array, a low index to the array and a high index to the array, designating a subarray, generates three random indexes in between low and high inclusive (let these be indexA, indexB and indexC), and the returns the corresponding index (either indexA, indexB or indexC) to the middle value of the three designated values array[indexA], array[indexB] or array[indexC]. A function randInt(a, b) is provided for you to call, which returns a random integer in between a and b, inclusive. **(Note: To make the problem a bit easier, there is no need to make sure that indexA, indexB and indexC are all distinct.)**

```
// Pre-condition: low and high are valid indexes into array with
//                high - low > 10
int getPartitionIndex(int array[], int low, int high) {









}

int randInt(int a, int b) {
    int base = ((rand()%32768)<<15) + (rand()%32768);
    return a + base%(b-a+1);
}
```