2. **Array Algorithms (25 pts, 7+6+6+6)**

For each of the following, write the algorithms in the form of bullet items (max 3-4 items), and give reasoning for the running time. A non-fastest algorithm will get at most half credit, and running time without reasoning will get no credit. None of the input arrays should be modified.

a) Two sorted arrays of length m and n respectively are to be merged into a new single sorted array of length $m+n$. Write the fastest algorithm. What is the exact number of item-to-item comparisons (NOT big O) in the worst case? Assume there are no duplicate or common items.

b) All integers in the range x to y ($x \leq y$) are to be found in a sorted integer array of length n. Assume that both x and y are in the array. Write the fastest algorithm. What is the worst case big O running time if there are k integers in the array within the range?

198:112 S18 Midterm Exam 1; netid: SM1841

c) Given an array A of integers of length n, *partial sums* of its items are to be computed in a new result array R of the same length: R[i] = sum of integers A[0] through A[i]. So, for instance, if A is [1,5,3,6,2], then R would be [1,6,9,15,17]. (Array A is NOT modified.) Write the fastest algorithm. What is the worst case big O running time?

d) An array of length n holds student scores on an exam. Scores are 0..100, in multiples of 5. You need to compute the number of students for each possible score (i.e. how many scored 0, 5, 10, 15 ... 100). Write the fastest algorithm (use extra storage space if needed). What is the worst case big O running time?

c) You are given the following (modified) sequential search code for searching in a list that is *sorted* in increasing order of values:

```
public boolean search(int[] A, int target) {
    for (int i=0; i < A.length; i++) {
        if (target == A[i]) return true;
    }
    if (target < A[i]) return false;
    return false;
}
```

For an array of length n, work out an exact formula in n (**not** big O) for the average number of target-to-arrayitem comparisons for successful searches. Assume equal probabilities for matching against any item. You don't need to simplify your answer down to a single term.

target-to-arrayitem comparisons for successful searches. Assume equal probabilities against any item. You don't need to simplify your answer down to a single term.

d) Consider a sorted array of 5 items on which searches are made, with probabilities of match distributed as 0.3,0.25,0.2,0.15,0.1. Would the average number of comparisons for success be better with binary search, or with sequential search applied on optimal rearrangement of the items? Show your work.