1. (4 points) Construct an optimal binary search tree for the keys A, B, C, D, E, F with respective search probabilities 0.22, 0.13, 0.21, 0.20, 0.08, and 0.16 by computing entries for the tables "A" and "r".

2. (4 points) Given a sequence of n numbers a1, a2, a3, . . . , an (some of them might be negative) stored in an array, we want to find two indices i <= j such that the sum of the numbers from ai to aj is maximum, among all possible i j pairs 1 <= i <= j <= n.
   2a) Write pseudocode to sum each contiguous subsequence (from ai to aj) and keep track of the maximum one. What is the runtime of your algorithm?
   2b) Now find an O(n) algorithm. Give pseudocode.

3. (5 points) Problem 15-6: Planning a company party - Professor Stewart is consulting for the president of a corporation that is planning a company party. The company has a hierarchical structure; that is, the supervisor relation forms a tree rooted at the president. The personnel office has ranked each employee with a conviviality rating, which is a real number. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to attend.

   Professor Stewart is given the tree that describes the structure of the corporation, using the left-child, right-sibling representation described in Section 10.4. Each node of the tree holds, in addition to the pointers, the name of an employee and that employee's conviviality ranking. Describe an algorithm to make up a guest list that maximizes the sum of the conviviality ratings of the guests. Analyze the running time of your algorithm.

4. (4 points) Let F be a text file consisting of 128 characters, each in the set A = {a; b; c; d; e; f; g; h}, and let T be a Huffman encoding tree for F. The height of T depends on the frequencies the characters of A in F.
   4a) Assign a frequency to each character of A such that the height of T is maximum.
   4b) Assign a frequency to each character of A such that the height of T is minimum.

   In each case, determine the length (number of bits) of the encoded file. Note that in each case, the sum of the frequencies of the characters must be 128.

5. (4 points) Optimal Merge Pattern - We saw from MergeSort that merging two sorted files together (one size m and one size n) took O(m+n) time. When merging multiple sorted files together, the merge can be accomplished by repeatedly merging sorted files in pairs. For example X1, X2, X3 are three sorted files of length 30, 20 and 10. There are multiple ways to merge them two-at-a-time (with the appropriate costs):

<table>
<thead>
<tr>
<th>(X1 merge X2) merge X3</th>
<th>X1 merge (X2 merge X3)</th>
<th>(X1 merge X3) merge X2</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost=50</td>
<td>cost=60</td>
<td>cost=40</td>
</tr>
<tr>
<td>total cost=110</td>
<td>total cost=90</td>
<td>total cost=60</td>
</tr>
</tbody>
</table>

   Different pairings require different amounts of computing time. How would you solve this problem to find the optimal way to merge the files two-at-a-time. Prove the required properties of the problem and for your solution approach.
6. (4 points) You are an algorithms consultant for a car share company, which provides temporary vehicles to people at a variety of fixed locations. The day before a user needs a car, he goes to a website and chooses the time and location of the pick-up, and the time and location of the drop-off. We need to make sure that enough cars are at each location at the start of the day, so that there is always at least one car available for every pick-up. However, it is expensive to place cars (overnight) at the various locations, so minimizing the total number of cars in circulation is an important goal. We want an efficient algorithm to process the vehicle requests so as to minimize the total number of cars that must be initially placed at the locations.

Formally, the problem is the following: you are given M locations (numbered 1 through M) and N tuples \((L, T, L', T')\) in some arbitrary order. Here \(L\) and \(T\) are location and time of pick up, and \(L'\) and \(T'\) are location and time of drop off. When a user picks up a car from a location, the number of cars at that location decreases by 1; when she drops off a car to a location, the number of cars at that location increases by 1. Give an algorithm that outputs the number of cars that should initially be circulated to each location subject to two constraints:
- every location always has a non-negative number of cars, and
- the total number of cars in circulation is minimized.

Make your algorithm as efficient as you can. You can assume everyone returns their car on time and to the correct location.