CS330 – Design and Analysis of Algorithms – Midterm

Name: _____

1. True/false

   (a) $n^2 + 10n + \sqrt{n} = O(n^3)$

   (b) $n^2 + 10n + \sqrt{n} = O(n^2)$

   (c) $n^2 + 10n + \sqrt{n} = O(n)$

   (d) $\log n = O(1)$

   (e) $\sqrt{\log n} = O(\sqrt{n})$

   (f) $(\log n)^{10} = O(\sqrt{n})$

   (g) $\left(\frac{5}{4}\right)^n = O(n^2)$

2. - Write down all permutations of A,B,C.

   - Write down all subsets of {A,B,C}.

   - Write down all combinations of 3 from {A,B,C,D}.

3. Write an algorithm with run-time complexity described by the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2$$

$$T(1) = 1$$

   Algorithm does not have to do anything useful.

4. Solve recurrence from the previous problem by drawing the recursion tree.

5. *Cryptarithm* is a puzzle in which digits in a proper arithmetic expression are substituted by letters (in case there are multiple instances of a digit all of them are substituted by the same letter). An example:

$$225 + 1238 = 1463$$

so after $1 \to A, 2 \to B, 3 \to C$, etc., we get

$$BBD + ABCE = AGFC$$

Write psedo code for brute-force algorithm that solves a given cryptarithm, i.e. given a cryptarithm find which digits correspond to which letters so that the resulting arithmetic expression is correct. Remember – I'm interested in overall structure of the algorithm rather than details, for example I'm not interested in how to check if a particular assignment of letters to digits can be tested – just assume you have a function that returns a boolean. But if you have a `for/while` loop – make sure to be precise explaining what kind of objects it traversing.

Psedo-code

What is the worst run-time complexity of your algorithm?

Can you use branch-and-bound optimization for this problem? Explain.

6. Longest Common Subsequence (LCS) Problem is as follows – given 2 strings find the longest subsequence of indices in the first and equal length subsequence in the second, so that characters at the corresponding position of the subsequences are the same. Mathematically: given $S1[1] \dots S1[n]$ and $S2[1] \dots S2[m]$ (note strings may have different length) find $I1[1] \dots I1[len]$ and $I2[1] \dots I2[len]$ so that $S1[I1[1]] = S2[I2[1]] \dots S1[I1[len]] = S2[I2[len]]$ and len is maximized. An example strings are

`string 1`	`d`	`d`	`a`	`e`	`l`	`a`	`g`	`g`	`o`
`match`			`*`		`*`		`*`		`*`
`index`	`1`	`2`	`3`	`4`	`5`	`6`	`7`	`8`	`9`
`match`	`*`	`*`			`*`	`*`			
`string 2`	`a`	`l`	`f`	`d`	`g`	`o`	`a`		

So maximum length subsequence is `algo` and corresponding subsequences are $3, 5, 7, 9$ and $1, 2, 5, 6$.

Discuss how to design branch-and-bound optimization. Make sure to specify how to calculate the bound, whether it is an upper or lower bound and how branch termination condition works. Assume your already have a backtracking algorithm that solves the problem: the exact details of how traversal is implemented are not important, but in each node of the search tree you have part of subsequences $I1[1] \dots I1[k]$ and $I2[1] \dots I2[k]$ that index equal characters.

7. Prove correctness of the following algorithm

```
alg(int A, int B) { //assume A,B>0
    R = 1;
    I = 2*A;
    while (I > A) {
        R = R * B;
        I = I - 1;
    }
    return R;
}
```

which calculates B^A