COP 3502 (Computer Science I) Final Exam 12/10/2015

Last Name: _____ , First Name: _____

1) (10 pts) Perform the following base conversions:

a) 1347_{10} converted to base 7.

_____

b) 4562_{8} converted to base 10.

_____

c) 10011101011101_{2} converted to base 16.

_____

d) 4567_{9} converted to base 7.

_____

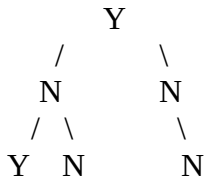
2) (10 pts) Use the iteration technique to determine a Big-Oh solution for the following recurrence relation:

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2, T(1) = 1$$

3) (11 pts) Write a function that takes in pointers to two sorted linked lists, combines them by rearranging links into one sorted linked list, effectively merging the two sorted lists into one, and returns a pointer to the new front of the list. Since no new nodes are being created and no old nodes are being deleted, your code should **NOT** have any mallocs or frees. Also, note that this destroys the old lists. If you try to print either listA or listB after calling merge, the lists are likely to print differently. (***Hint: This is probably much easier to do recursively.***) Fill in the function prototype provided below and use the struct provided below:

```
typedef struct node {
    int data;
    struct node* next;
} node;

node* merge(node* listA, node* listB) {




}
```

4) (10 pts) A company has a hierarchy that can be represented as a binary tree, with the CEO at the root, which we'll call level 1. The two VPs under the CEO are level 2. Direct subordinates to the VPs are level 3, and so forth. This company has a rather strange voting system. For each person in the company on level L, her vote counts as 1/L of the CEO's vote. Thus, in general, if a person on level L votes in favor of a potential new company policy, their vote counts 1/L points to the total. If they vote no, their vote counts as 0. A policy passes if the sum of the votes as described above strictly exceeds 1/2 of the total sum obtained if everyone were to vote yes. Consider the following company structure with the votes shown:



The two yes votes add a total of 1 (CEO) and 1/3 (bottom left) for a total of 4/3. If everyone were to vote yes, we'd have a sum of 1 + 2(1/2) (two people on level 2) + 3(1/3) (three people on level three) = 3. Since 4/3 is not strictly greater than 3/2, the measure does not pass.

Write two functions that take in the company vote tree (like the one depicted above) representing the vote on a single issue and the level of root node and calculate the maximum total sum if everyone voted yes and the actual total sum of the people voting yes, for that subtree. We store a capital 'Y' to indicate a yes vote and a capital 'N' to indicate a no vote. A division of these two values will determine if a vote passes. This wrapper code has already been provided for you.

```
typedef struct treenode {
    char vote;
    struct treenode* left;
    struct treenode* right;
} treenode;

int votePasses(treenode* root) {
    double total = maxTotal(root, 1);
    double votes = voteTotal(root, 1);
    return votes/total > 0.5 + 1e-9;
}

double maxTotal(treenode* root, int level) {









}
```

```
double voteTotal(treenode* root, int level) {









}
```

5) (7 pts) One way to calculate the cube root of a positive number is to run a binary search between a number known to be less than the desired cube root and a number known to be greater than the desired cube root. Write a ***recursive*** function that returns an approximation within 10^{-9} to the cube root of a double x, where the cube root is known to be in between a double low and a double high. A wrapper function has been written for you that calls your recursive function. Remember that the function $f(x) = x^3$ is a monotonically increasing function. (Hint: Your base case is when low and high are within 10^{-9} of each other!!!)

```
#define EPSILON 1e-9

double mycuberoot(double x) {
    double max = 1;
    if (x > max) max = x;
    return reccuberoot(x, 0, max);
}

double reccuberoot(double x, double low, double high) {









}
```

6) (7 pts) Complete the program below so that it prints out all the non-empty subsets of the array of candies given.

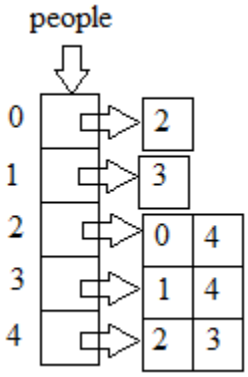
```
#include <stdio.h>

#define NUMCANDIES 8
#define SIZE 20

int main() {

    int i, j;
    char candies[NUMCANDIES][SIZE] = {"SNICKERS", "MANDMS", "REECES",
                "SKITTLES", "OHHENRY", "BABYRUTH", "HEATHBAR", "KITKAT"};

    for (i=1; i < _____________________________; i++) {

        for (j=0; j < _____________________; j++) {


            if ( ____________________________________________________ )
                printf("%s ", candies[j]);
        }
        printf("\n");
    }
    return 0;
}
```

7) (5 pts) 100,000 thousand searches on a database with $n = 2^{16}$ records takes 60 milliseconds. A single search on this database with n records executes in O(lg n) time. Consider a situation where the database size increased to $n = 2^{20}$. How long would 500,000 searches on this database take?

8) (12 pts) An efficient way to store data about the relationships amongst a group of n people is as follows: give each person a unique label from 0 to n-1. For each person, create a dynamically sized array, the size of the number of acquaintances that person has. For example, if person 0 has 3 acquaintances, then their array would be size three and each item in this array would be the number of one of their acquaintances. Since the people are labeled 0 to n-1, we can store each person's array, in an array itself! Consider an example with 5 people, where we have the following acquaintance pairs:

(0, 2), (1, 3), (2, 4), (3, 4)

We would store this in an array of size 5, where each item is an array as follows:



Write a function that takes in a file pointer to a file that stores data about a set of acquaintances, dynamically allocates the array of arrays described above and returns a pointer to that array of arrays. The file format is as follows:

The first line contains a single positive integer n, the number of people described in the file. The following n lines contain information about the acquaintances of each of person 0 through person n-1. Each of these lines starts with a positive integer n_i, the number of acquaintances of person i. The following n_i values are the numbers of the people that person i is acquainted with, in increasing order. It's guaranteed that the data is consistent. If person i is acquainted with person j, then person j will be acquainted with person i. The input file storing the data shown in the previous picture is as follows:

```
5
1 2
1 3
2 0 4
2 1 4
2 2 3
```

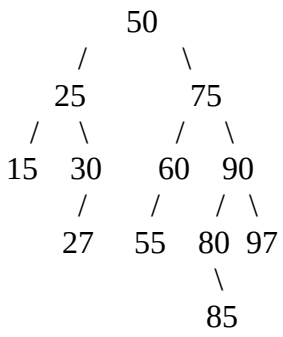
Please fill in the function prototype given on the next page. (Note: To actually implement this in a meaningful way, we'd have to store the value of n and the sizes of each of the arrays so that we could properly iterate through the structure. I haven't asked you to do these tasks in order to make the solution shorter.)

```
int** readgraph(FILE* ifp) {




}
```

9) (6 pts) During a Merge Sort of 8 items, the merge function gets called 7 times. Show the result of the array after each of the merges, as they occur in sequential order, on the array shown below. (The last row has been filled in for you since after the last merge, the array is sorted.)

Initial	18	6	5	13	9	17	14	1
merge 1								
merge 2								
merge 3								
merge 4								
merge 5								
merge 6								
Last	1	5	6	9	13	14	17	18

10) (5 pts) Show the final result of deleting the node storing 15 from the AVL tree shown below. Draw a box around your answer.



11) (5 pts) Consider inserting the following items into an empty minimum heap, in this sequence: 18, 12, 9, 22, 13, 17, 6, 47, 15, and 8. Show the final contents of the heap, as they are stored in a 1-based array. You may draw tree pictures of the varying stages of the heap for partial credit, but for full credit you must fill in the array provided below.

Index	1	2	3	4	5	6	7	8	9	10
Value										

12) (5 pts) Using separate chaining hashing, and the hash function $f(x) = (x^2 + 3x + 7) \% 19$, draw a picture of what the structure would look like after adding the following values, in this order: 3, 4, 7, 8, 2, 9, 14, 1, 15, and 12. Assume that we insert new items to the ***front*** of the appropriate linked list.

13) (6 pts) Determine a closed form for the following summation in terms of n:

$$\sum_{i=n+1}^{n^2} i$$

(Note: An extra credit point will be given if you can fully factor your final answer!)

_____

14) (1 pt) On December 10th, 1948 the United Nations General Assembly adopted the Universal Declaration of Human Rights. As a consequence, December 10th is Human Rights Day. By what acronym is the United Nations better known?

_____