Fall 2021 COP 3502 Section 2 Final Exam - Part A Solution

1) (10 pts) For a particular video game, each user has a name (letters only) and a number of points (non-negative integer less than 10^9). Imagine reading in from standard input, information about n users and storing that information in an **array of pointers to user**, where user is the struct shown below. The format of the information being inputted is as follows:

1. First line has a single positive integer, *n*, representing the number of users.
2. Each user follows, with the information for each user being on three lines.
    a. The first line for the i^{th} user has a single positive integer L_i, representing the length of the i^{th} user's username

    b. The second line for the i^{th} user has a string of L_i letters, representing the name of the i^{th} user.

    c. The third line for the i^{th} user has an non-negative integer, p_i, representing the number of points the i^{th} user has.

Here is a short sample input:

```
4
13
SUPERCRAZYMAN
2000
11
WonderWoman
6789
17
JakeFromStateFarm
12
18
FloFromProgressive
49
```

Complete the function below to read in this information and return a pointer to an array of pointers to struct storing this information. The struct storing a user is as follows:

```
typedef struct user {
    int nameLen;
    char* name;
    int points;
} user;
```

In particular, your function is responsible for dynamically allocating all the necessary memory AND reading in the information into the array of pointers to struct and returning a pointer to the resulting array. **Make sure to allocate exactly the necessary space for the pointer name for each user struct.**

In order for this function to be useful, a pointer to the variable storing the size of the array (first piece of information being read in) is the only input parameter to the function. Please read the first item into the variable pointed to by this pointer.

```
user** readUsers(int* ptrNumUsers) {

    scanf("%d", ptrNumUsers);                              // 1 pt
    user** res = malloc(sizeof(user*)* (*ptrNumUsers)); // 2 pts

    for (int i=0; i<(*ptrNumUsers); i++) {                 // 1 pt
        res[i] = malloc(sizeof(user));                     // 1 pt
        scanf("%d", &(res[i]->nameLen));                   // 1 pt

        // 2 pts
        res[i]->name = malloc(sizeof(char)*(res[i]->nameLen+1));

        scanf("%s%d", res[i]->name, &(res[i]->points)); // 2 pts
    }

    return res; // 0 pts

}
```

2) (8 pts) An arithmetic sequence is a sequence of numbers such that the difference between each pair of successive terms it the same. For example, 4, 7, 10, 13, and 16 is an arithmetic sequence of length 5 with a starting term of 4 and a common difference of 3. Write a **recursive function** which takes in the first term in an arithmetic sequence, the common difference of the sequence and the number of terms in the sequence and returns the sum of the sequence.

```
int sumArithSeq(int firstTerm, int diff, int nTerms) {


    if (nTerms == 0) return 0;    // 2 pts

    return firstTerm +
           sumArithSeq(firstTerm+diff, diff, nTerms-1);

    // 1 pt return, 1 pt add firstTerm (or last), 1 pt rec call
    // 2 pts new first term, 1 pt diff, 2 pts new # terms

}
```

3) (10 pts) The following function solves a problem, assuming that the input array is sorted:

```
// Pre-condition: sortedArray is a sorted from smallest to
//      largest, and has n values.
int whatDoesItDo(int* sortedArray, int n, int value) {

    int a = 0, b = n-1;
    while (a < b) {
        if (sortedArray[a]+sortedArray[b] == value) return 1;
        if (sortedArray[a]+sortedArray[b] < value)
            a++;
        else
            b--;
    }

    return 0;
}
```

(a) (4 pts) As specifically as possible, in English, explain what problem this function solves. Any explanation which tries to convert the code literally (then add one to a, etc.) will get **NO CREDIT.**

It returns 1 if and only if there exist 2 different numbers (numbers stored in different indexes in sortedArray) which add up to exactly value. It returns 0 if no two numbers exist.

Grading: 4 pts if clearly written in English, 0 pts if code is translated, give partial as needed.

(b) (4 pts) If the input array isn't sorted, then this function will NOT properly solve the problem designated in part (a) in all cases. Give a specific case (array, length of the array and value) where the solution to the problem is 1 but the function as its written returns 0.

array: 3, 2, 8, 1 n: 4 value: 3

The code sees that the first sum it calculates, 3+1, is greater than the target value of 3, so it decrements the high index, next adding 3 + 8. Again, this is too big, so this high index gets decremented, trying 3 + 2. Since this is too big, the high index gets decremented again and the function returns 0. But, the values in index 1 and index 3 (2 and 1, respectively), do add up to 3.

Grading: 2 pts for ANY example that clearly states the array, n and value, 2 more points if it's a counter-example (no need for them to explain, you can just trace it.)

(c) (2 pts) What is the run time of this function, in terms of the input parameter, n? Briefly justify your answer.

O(n), in each loop iteration the difference between a and b decrements by 1. Since this difference starts at n-1, the maximum number of loop iterations is n-1. Finally, there is O(1) work done inside the loop, which means the total run time is O(n). **(Grading: 1 pt ans, 1 pt justification)**

Fall 2021 COP 3502 Section 1 Final Exam - Part B Solutions

4) (10 pts) Imagine storing a string of letters in a linked list and building a set of string functions for this type of storage. Complete the function strCompare below which should operate the exact same way that strcmp behaves in the string class. Namely, if the string represented by the linked list pointed to by str1 comes lexicographically before the string represented by the linked list pointed to by str2, return a negative integer. If the string represented by the linked list pointed to by str1 comes lexicographically after the string represented by the linked list pointed to by str2, return a positive integer. If the strings are equal, return 0. The struct definition is provided below, along with the function prototype. You may assume that all characters are lowercase letters for ease of implementation. **(Hint: For ease of implementation, write recursively!)**

```
#include <stdio.h>
#include <stdlib.h>

typedef struct str {
    char letter;
    struct str* next;
} str;

int strCompare(str* str1, str* str2) {


    if (str1 == NULL && str2 == NULL) return 0; // 2 pts
    if (str1 == NULL) return -1;                // 1 pt
    if (str2 == NULL) return 1;                 // 1 pt
    if (str1->letter != str2->letter)           // 2 pts
        return str1->letter - str2->letter;     // 2 pts
    return strCompare(str1->next, str2->next);  // 2 pts

}
```

Grading Note: Please map the iterative solution, which is longer, accordingly. Also, many will do the last if as an if-else (maybe return -1 or return 1), which is just fine.

5) (12 pts) Solve the following recurrence relation (solve for a direct solution to T(n) in terms of a Big-Oh bound) in two ways: (a) Use the Master Theorem, (b) Use the Iteration Technique. (Note: Knowing what the end answer should be via step (a) should help you discover mathematical errors in part (b).)

$$T(n) = 3T\left(\frac{n}{2}\right) + n^2 \text{ (if n > 1)}$$
$$T(1) = 1$$

(a) Use Master Theorem

A = 3, B = 2, k = 2, since $B^k = 2^2 = 4$ and A = 3, we're in the last case with run time **$O(n^2)$.**

Grading: 2 pts all or nothing

(a) Use Iteration Technique

Let's work out the first three iterations:

$$\boldsymbol{T(n) = 3T\left(\frac{n}{2}\right) + n^2}$$
$$T(n) = 3(3T\left(\frac{n}{4}\right) + \frac{n^2}{4}) + n^2$$
$$\boldsymbol{T(n) = 9T\left(\frac{n}{4}\right) + n^2(1 + \frac{3}{4})}$$
$$T(n) = 9(3T\left(\frac{n}{8}\right) + \frac{n^2}{16}) + n^2(1 + \frac{3}{4})$$
$$\boldsymbol{T(n) = 27T\left(\frac{n}{8}\right) + n^2(1 + \frac{3}{4} + \frac{9}{16})}$$

After k steps we have: $\boldsymbol{T(n) = 3^kT\left(\frac{n}{2^k}\right) + n^2 \sum_{i=0}^{k-1}(\frac{3}{4})^i}$.

Plug in $k = \log_2 n$. (We get this by setting $n/2^k = 1$, since we know T(1).)

$$\boldsymbol{T(n) < 3^{log_2 n}T(1) + n^2\left(\frac{1}{1-3/4}\right) = n^{log_2 3} + 4n^2 = O(n^2)}$$

Note: We've substituted the summation with the corresponding infinite sum that is slightly more than the original sum, hence the < sign. Also, we used a log identity to simplify the first term and note that $\log_2 3 < \log_2 4 = 2$, to determine that the second term is the dominating term.

Grading: 1 pt first first iteration
2 pts for second iteration
2 pts for third iteration
2 pts for after k steps
1 pt to plug in appropriate k
2 pts to complete it

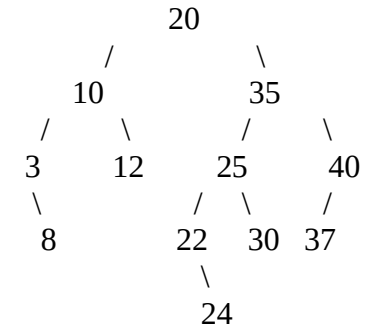
Fall 2021 COP 3502 Section 1 Final Exam - Part C Solution

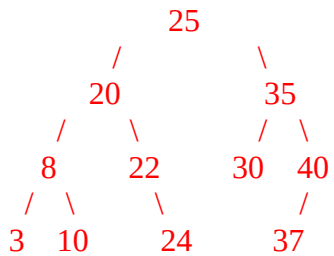
6) (5 pts) Consider tracing through a Merge Sort of the following array. Show the state of the array after each merge, in the order in which the merges occur. For clarity, the result after the first merge and the result after the last merge are included.

Initial Array	12	3	9	8	16	13	2	14
After 1st Merge	3	12	9	8	16	13	2	14
After 2nd Merge	3	12	8	9	16	13	2	14
After 3rd Merge	3	8	9	12	16	13	2	14
After 4th Merge	3	8	9	12	13	16	2	14
After 5th Merge	3	8	9	12	13	16	2	14
After 6th Merge	3	8	9	12	2	13	14	16
After 7th Merge	2	3	8	9	12	13	14	16

Grading: 1 pt per line, line must be perfectly correct to get the point

7) (7 pts) Show the result of deleting 12 from the AVL Tree shown below. Put a box around your final answer.





Grading: 1 pt 25 at root, 1 pt 20 to left, 1 pt 35 to right, 1 pt for each subtree attached

If they correctly rebalanced the 3, 8, 10 but did nothing else give 2 out of 7.
If they picked the wrong root give a max of 3 out of 7.

8) (10 pts) Write a function that takes in a pointer to a binary tree node which is the pointer to the root of a **binary search tree** and prints out the values stored **in the leaf nodes** in **reverse sorted order**, from largest to least. (Hint: The code represents an edit to one of the usual tree traversals, with an extra base case for printing.) Please use the struct provided below.

```
typedef struct treenode {
    int data;
    struct treenode* left;
    struct treenode* right;
} treenode;

void printLeafNodesReverse(treenode* root) {

    if (root == NULL) return;                        // 2 pts

    printLeafNodesReverse(root->right);              // 2 pts
    if (root->left ==NULL && root->right == NULL) { // 2 pts
        printf("%d ", root->data);                   // 2 pt
        return;                                      // 0 pts
    }
    printLeafNodesReverse(root->left);       // 2 pts



}
```

Fall 2021 COP 3502 Section 2 Final Exam - Part D Solutions

9) (5 pts) Convert 34564_7 to base 9.

$34564_7 = 3 \times 7^4 + 4 \times 7^3 + 5 \times 7^2 + 6 \times 7^1 + 4 \times 7^0$
$= 3 \times 2401 + 4 \times 343 + 5 \times 49 + 42 + 4$
$= 7203 + 1372 + 245 + 46$
$= 8866$, in base 10

9 | 8866
9 | 985 R 1
9 | 109 R 4
9 | 12 R 1
9 | 1 R 3

$34564_7 = \mathbf{13141_9}$

Grading: 2 pts sum of terms, 1 pt to get to 8866, 2 pts process and answer to 13141.

10) (6 pts) For accreditation, all computer science students in a class are tested on 20 skills, numbered 0 through 19. The accreditation committee wants to the number of skills that ALL of the students have successfully completed. The first input to the function is an array containing integers, all in between 0 and 2^{20}-1, where each integer represents the skills of a single student. The second input to the function is the length of the input array. In particular, if bit k is set to 1 in the value student[i], then that means that student i has completed skill i. Complete the function below to complete the task. You may call the countBits function provided in your function.

```
int numSkillsAllHave(int student[], int n) {

    int res = student[0];           // 1 pt
    for (int i=1; i<n; i++)         // 1 pt
        res &= student[i];          // 2 pts
    return countBits(res, 20);      // 2 pts
}

// Returns the number of bits in mask set to 1 out of the first
// n least significant bits.
int countBits(int mask, int n) {
     int res = 0;
     for (int i=0; i<n; i++)
          if ((mask & (1<<i)) != 0)
              res++;
     return res;
}
```

11) (8 pts) An n-digit sum divisible number is one such that for each prefix of k digits, the sum of those digits is divisible by k, for each value of k, ranging from 1 to n. For example, 372495 is a 6-digit sum divisible number because 3 is divisible by 1, 3+7 is divisible by 2, 3+7+2 is divisible by 3, 3+7+2+4 is divisible by 4, 3+7+2+4+9 is divisible by 5 and 3+7+2+4+9+5 is divisible by 6. Complete the code below so that it prints out each 7-digit sum divisible number that doesn't use repeat digits and doesn't start with a 0. (Note: we store a number digit by digit.)

```
#include <stdio.h>

void go(int digits[], int used[], int curSum, int k, int n);
void print(int digits[], int n);

int main(void) {
    int digits[7];
    int used[10];
    for (int i=0; i<10; i++) used[i] = 0;
    go(digits, used, 0, 0, 7);
    return 0;
}

void go(int digits[], int used[], int curSum, int k, int n) {

    if (k == n) {
        print(digits, n);
        return;
    }

    int start = (k > 0) ? 0 : 1;

    for (int i=start; i<10; i++) {

        if ( used[i] ) continue;              // 1 pt

        if ( (curSum+i)%(k+1) != 0 ) continue;    // 3 pts
        used[i] = 1;
        digits[k] = i ;                       // 1 pt
        go(digits, used, curSum + i  , k+1, n); // 2 pts
        used[i] = 0 ;                         // 1 pt
    }

}

void print(int digits[], int n) {
    for (int i=0; i<n; i++) printf("%d", digits[i]);
    printf("\n");
}
```

12) (8 pts) You have discovered that if you study for a final exam for ***t*** hours, the score you will obtain on the final is expressed by the function $f(t) = \sqrt{t} + 2^{\frac{t}{10}}$. You have determined that you need a score of ***desiredScore*** to obtain an A in the course. Complete the function below so that returns the number of hours (real number) you will need to study to obtain exactly the score of ***s*** (within a reasonable tolerance), so that you don't "over" or "under" study.

```
#include <math.h>

double amtToStudy(double desiredScore) {


    double low = 0, high = desiredScore*desiredScore ;

    for (int i=0; i<60; i++) {

        double mid = (low+high)/2;                  // 1 pt

        double score = sqrt(mid) + pow(2, mid/10);  // 3 pts

        if (score < desiredScore)                   // 1 pt
            low = mid;                              // 1 pt
        else                                        // 1 pt
            high = mid;                             // 1 pt
    }

    return low;
}
```

Grading Note: if they swap the order of the if (so assign low when they should have assigned high and vice versa), take off 2 points.

13) (1 pt) What holiday is celebrated by many the day after Christmas Eve? **Christmas**
(Give to all)