COP 3502 – Spring 2012
Exam #1 *(Solutions)*
2/17/2012

Exam 1 Comments:

As mentioned in class, my intention with this first exam is to have, at most, one hard question, which is almost always going to be the recursive coding question or the linked-list coding question, simply due to the fact that you have to actually think instead of regurgitate. Other than that, I wanted all other questions to be "workable"; meaning, every student who did an average amount of studying could successfully solve the problem. All test questions were straightforward, and anyone who truly studied and understood the PPT slides, lab questions, and programs would have done very well.

Statistics:

# of Exams Taken:	**216**
Time Stats:	**1st student finished at the 35 minute mark** **22 students finished with 15 minutes remaining** **43 students finished with 10 minutes remaining**
Average Grade:	**81.37 (which is a HIGH average)**
# of Grades >= 100:	**29**
# of Grades >= 90:	**83**
# of Grades >= 80:	**140**

So out of 216 students, 140 scored higher than 80. Yeah, that is HUGE. I'd love to say this is completely due to the awesomeness of (a) the students and (b) the instructor; that would be great. Most likely, there are other factors at play.

Regarding the length, I think it was fair, and the stats "Time Stats" agree. Ideally, the first student to leave (assuming he/she did well) finishes around the midway point. A few more trickle in. And with five to ten minutes left, it is a good sign when one-third of the class has finished. Finally, when the time was up, only around 20-30 students (less than 10% of the class) were still working (or reading over their answers). These are good indicators as to the length and difficulty of an exam. So although some may feel the exam was long, I do feel the length was appropriate for this class and certainly workable by those students that invested enough time in studying. Now to the solutions…

1) (10 pts) **Dynamic Memory Allocation.** The following struct can be used to store information about a Super Market:

```
struct SuperMarket {
  char SuperMarketName[30];
  int *itemNumbers;
  double *itemPrices;
};
```

The last two members of this struct are meant to be pointers to dynamically allocated arrays of ints and doubles, respectively. The first will be an array of item numbers, each stored as an `int`, representing the item numbers for all items in the store. The second will be an array of prices, each stored as a `double`, representing the prices of those respective item numbers.

The segment of code below asks the user to enter one positive integer, representing the number of items in the store, and this value is saved into "**k**".

Write a segment of code that **FULLY** allocates space for ONE struct SuperMarket (named below as mySuperMarket), where this struct has space to store exactly `k` item numbers and `k` item prices. Note: You don't need to initialize the allocated space; **you just need to allocate it**.

The beginning part of the code segment is given to you. Declare any extra variables you need.

```
int k;
struct SuperMarket *mySuperMarket;
printf("How many items will be in your store?\n");
scanf("%d", &k);

mySuperMarket = (struct SuperMarket*)malloc(sizeof(struct
                                                SuperMarket));
mySuperMarket->itemNumbers = (int*)malloc(sizeof(int)*k);
mySuperMarket->itemPrices = (double*)malloc(sizeof(double)*k);
```

COMMENTS: This question was pretty easy for someone who understood the memory allocation as done in Program 2. This should have taken two to three minutes if you understood memory allocation, but should have been difficult otherwise. You see that we first allocate memory for the ONE node of mySuperMarket, and then we allocate memory for the dynamic array within that ONE struct. About 60% got this completely correct.

2) (4 pts) **Memory Management.** Free the memory allocated to `mySuperMarket` in the previous question.

```
free(mySuperMarket->itemNumbers);
free(mySuperMarket->itemPrices);
free(mySuperMarket);
```

3) (14 points) **Linked Lists.** Write a function that operates on an existing linked list of integers. Your function should **insert** a new node containing the data value 7 **after every node** that contains the data value 5. Make use of the `listNode` struct and function header below.

```
struct listNode {
     int data;
     struct listNode* next;
};

void list_57(struct listNode* head)
{

     struct listNode *help_ptr = head;
     struct listNode *temp;

     while (help_ptr != NULL)
     {
          if (help_ptr->data == 5)
          {
               temp = malloc(sizeof(struct listNode));
               temp->data = 7;
               temp->next = help_ptr->next;
               help_ptr->next = temp;
          }
          help_ptr = help_ptr->next;
     }

}
```

COMMENTS: about 40 students got this one perfect! The problems were all over the place and hard to list out. One point was noticed: for many who got it right, we found a drawn out picture of the nodes and what was going on! GREAT job! That helps you get it and helps to figure out which pointers need to be stored where.

4) (8 points) **Practical Problem in Algorithm Analysis.** Algorithm D runs in $O(n^3)$ time. For an input of size 2, the running time is 2 milliseconds. How long will Algorithm D take to run on an input size of 5? **You MUST show your work**. An answer without work will get only 1 pt.

We start by finding our constant, c.

$T(n) = c*n^3$, so now use the given values to find c.
$T(2) = c*2^3 = 2$
$T(2) = c*8 = 2$, solve for c and we get $c = 2/8 = 1/4$

Now use that c and the new input size, 5, to find the new running time.
$T(5) = (1/4)*5^3 = (1/4)*125 = 31.25$ milliseconds

5) (14 points) **Recursion.** Write a RECURSIVE function, `printTriangle`, that prints a triangle of asterisks whose length and height are of size, *n*. For example, if `printTriangle(5)` is called from main, the following will print to the screen:

```
* * * * *
* * * *
* * *
* *
*
```

HINT: Although you will use recursion to solve this, there can be a `for` loop inside the recursive function that you will write.

Please use the function header provided below for your function:

```
void printTriangle(int n)
{
     int i;

     for (i = 0; i < n; i++);
          printf("* ");
     printf("\n");

     if (n > 1)
          printTriangle(n-1);

}
```

COMMENTS: approx. 50% got this correct. Minor errors included some infinite loops and printing at wrong spots, but most had a good idea of what to do.

6) (8 points) **Practical Problem in Algorithm Analysis.** Algorithm Q runs in $O(n \log_2 n)$ time. For an input of size 16, the running time is 8 milliseconds. How long will Algorithm Q take to run on an input size of 64? **You MUST show your work**. An answer without work will get only 1 point.

We start by finding our constant, c.

T(n) = c*nlogn, so now use the given values to find c.
T(16) = c*16log16 = 8
T(16) = c*16*4 = 8, solve for c and we get c = 8/64 = 1/8

Now use that c and the new input size, 64, to find the new running time.
T(64) = (1/8)*64log64 = (1/8)*64*6 = 8*6 = 48 milliseconds

7) (12 points) **Recursion.** Consider the following TWO recursive functions:

```
void recursiveFuncA(int n) {
     printf("%d ", n);
     if (n > 0)
          recursiveFuncA(n-8);
}
```

What would be printed by calling `recursiveFuncA(24)`?

Put your answer in the box below:

24 16 8 0

```
void recursiveFuncB(int n) {
     if (n > 0)
          recursiveFuncB(n-8);
     printf("%d ", n);
}
```

What would be printed by calling `recursiveFuncB(24)`?

Put your answer in the box below:

0 8 16 24

COMMENTS: around 60% got this correct. Common errors were on recursiveFuncB. Many students printed the zero and didn't go back and finish (print) the previous recursive calls.

8) (12 points) **Summations.** Determine a closed-form solution for the following summation in terms of n. **You MUST show your work.**

$$\sum_{j=n-10}^{n}\left(4\sum_{i=1}^{3j}2\right)=$$

$$\sum_{j=n-10}^{n}\left(4\sum_{i=1}^{3j}2\right)=\sum_{j=n-10}^{n}\left(8\sum_{i=1}^{3j}1\right)=\sum_{j=n-10}^{n}8*3j=\sum_{j=n-10}^{n}24j$$

$$\sum_{j=n-10}^{n}24j=24\sum_{j=n-10}^{n}j=24\left(\sum_{j=1}^{n}j-\sum_{j=1}^{n-11}j\right)$$

$$\mathbf{24}\left(\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}}\boldsymbol{j}-\sum_{\boldsymbol{j=1}}^{\boldsymbol{n-11}}\boldsymbol{j}\right)=\mathbf{24}\left(\frac{\boldsymbol{n(n+1)}}{\mathbf{2}}+\frac{\boldsymbol{(n-11)(n-10)}}{\mathbf{2}}\right)$$

COMMENTS: around 80% got it right. GREAT job! Common mistake was simple forgetting the limit change rule.

9) (8 points) **Algorithm Analysis.** Determine the **Big-O** running time of the following code fragment. Do NOT use summations for this problem. Simply analyze the code and give the **Big-O** running time in terms of the variable n. **YOU MUST EXPLAIN YOUR ANSWER**. An answer without explanation gets only 1 point.

```
int i, j, someValue = 0;
for(i = 1; i < 14n^2; i++) {
     for(j = 1; j < 42n^5 j++) {
          someValue = someValue + 7;
     }
}
```

Outer loop iterates n^2 times (from i = 1 to $14n^2$).
For each iteration of the outer loop, the inner loop iterates n^5 times (from j = 1 to $42n^5$).
For each iteration of the inner loop, there is 1 operation (constant work).
So we have $14n^2*42n^5 = 14*42* n^7$, or $O(n^7)$.

COMMENTS: around 80% got it right. Not much else to say here.

10) (10 points) **Algorithm Analysis.** Write, but DO **NOT** solve, a summation that describes the number of multiplications performed by the following code fragment, in terms of n.

```
weirdSum = 0;
for(i = 5; i <= n + 155; i++) {
     weirdSum = weirdSum + 5 - i;
     for(j = i - 5; j < i + i + 25; j++) {
          weirdSum = weirdSum * n - j * 3 + i * j;
     }
}
```

$$\sum_{i=5}^{n+155}\left(\sum_{j=i-5}^{2i+24}3\right)$$

COMMENTS: around 80% got it right. Not much else to say here.

11a) (2 points) **Pointer/Address.** In one word, what is a pointer? (Hint: the answer starts with an "a" and ends with an "s" and has seven letters total)

Address

11b) (3 points) **Freebie.** What is the best movie you've ever seen? **Must answer for credit**.
Any answer counts
(well, except for Gigli…that automatically results in a "-10" for this question!)