2016 Spring COP 3502 Exam #1 Solutions - Dynamic Memory Allocation (20 pts)

Date: 2/18/2016

Consider a video game with multiple levels, where each level can be represented with a two-dimensional array of integers. For the following questions you will read in information about the number of levels and the dimensions of each of the levels and allocate the appropriate amount of space to store the information.

1) (5 pts) Complete the blanks in the segment of code below to read in a single integer from standard input into the variable numLevels and then allocate that many pointers to two dimensional arrays. (You will not allocate the space for any of the 2D arrays yet, just for the pointers to those arrays.)

```
int numLevels;

scanf("%d", &numLevels); // Grading: 2 pts

int*** game = malloc(numLevels*sizeof(int**)) ; // Grading: 3 pts
```

2) (13 pts) Complete the blanks in the segment of code below so that it reads in numLevels pairs of integers, r_i and c_i ($0 \leq i \leq$ numLevels), where the i^{th} pair are the dimensions of the i^{th} level, and dynamically allocates space for each level using the appropriate set of mallocs.

```
int i, j, numRows, numCols;
for (i=0; i<numLevels; i++) {

    scanf("%d%d", &numRows, &numCols);

    game[i] = malloc(numRows*sizeof(int*)) ; // Grading: 5 pts

    for (j=0; j<numRows; j++)                // Grading: 3 pts

        game[i][j] = malloc(numCols*sizeof(int)) ; // Grading: 5 pts

}
```

3) (2 pts) If the input to the segments of code above was

```
3
2 10
8 5
3 6
```

how many malloc statements would be executed by the segments of code above?

17 (Grading: 2 pts all or nothing)

2016 Spring COP 3502 Exam #1 Solutions - Mathematical Tools (30 pts)

Date: 2/18/2016

4) (10 pts) Determine a closed for solution to the following summation, in terms of n:

$$\sum_{i=2n+1}^{3n} (2i-1)$$

$$\sum_{i=2n+1}^{3n} (2i-1) = 2\left(\sum_{i=1}^{3n} i - \sum_{i=1}^{2n} i\right) - (3n - (2n+1) + 1)$$
$$= 2\left(\frac{3n(3n+1)}{2} - \frac{2n(2n+1)}{2}\right) - (3n - 2n - 1 + 1)$$
$$= 3n(3n+1) - 2n(2n+1) - n$$
$$= 9n^2 + 3n - 4n^2 - 2n - n$$
$$= 5n^2$$

Grading: 2 pts to split summation into four pieces, 2 pts to evaluate the sums of constants, 2 pts for sum of i to 3n, 2 pts for sum of i to 2n, 2 pts simplification. (If bounds on any sum are off by one but rest of the work is consistent, just take off 1 pt per location with incorrect bounds.)

5) (10 pts) Consider the process of solving the recurrence relation $T(n) = 4T\left(\frac{n}{2}\right) + n^3$ using the iteration technique. Carry this process out to the third iteration where you get an equation of the form $T(n) = 64T\left(\frac{n}{A}\right) + \frac{B}{C}n^3$, where *A*, *B*, and *C* are each integers, determining the values of A, B and C. (Partial credit will be given for finding the correct equation at the second iteration.)

$$T(n) = 4T\left(\frac{n}{2}\right) + n^3$$

$$T(n) = 4(4T\left(\frac{n}{4}\right) + (\frac{n}{2})^3) + n^3$$

$$T(n) = 16T\left(\frac{n}{4}\right) + 4 \times \frac{n^3}{8} + n^3 = 16T\left(\frac{n}{4}\right) + \frac{n^3}{2} + n^3 = 16T\left(\frac{n}{4}\right) + \frac{3n^3}{2}$$

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

$$T(n) = 64T\left(\frac{n}{8}\right) + 16 \times \frac{n^3}{64} + \frac{3n^3}{2} = 64T\left(\frac{n}{8}\right) + \frac{n^3}{4} + \frac{3n^3}{2} = 64T\left(\frac{n}{8}\right) + \frac{7n^3}{4}$$

$$T(n) = 64T\left(\frac{n}{8}\right) + \frac{7n^3}{4}$$

(So A = 8, B = 7, and C = 4)

Grading: Full credit if they A, B and C. 5 pts for arriving at 2nd iteration. Within each iteration 2 pts for plugging in initially, 3 pts for simplifying

6) (10 pts) An $O(n^2)$ algorithm takes 100 ms with an input size of n = 50000. Roughly, how long, in ms, will the algorithm take on an input of size n = 80000?

Let T(n) be the run time. Then $\boldsymbol{T(n) = cn^2}$.

$$\boldsymbol{T(50000) = c(50000)^2 = 100ms}$$

$$\boldsymbol{c = \frac{100ms}{50000^2}}$$

$$\boldsymbol{T(80000) = c(80000)^2 = \frac{100ms}{50000^2} \times 80000^2 = \frac{100ms \times 64}{25} = 4 \times 64\, ms = 256ms}$$

Grading: 3 pts for solving for c, 3 pts for plugging in n = 80000 and c, 4 pts for simplifying.

2016 Spring COP 3502 Exam #1 Solution - Recursion (30 pts)

Date: 2/18/2016

7) (10 pts) The 3n+1 problem is as follows: Given a positive integer, *n*, calculate the next number in the sequence by dividing *n* by 2, if *n* is even, or calculating 3*n* + 1, if *n* is odd. Continue the sequence until the number 1 is generated. If n = 7, the sequence created is as follows:

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

Thus, if we start with n = 7, the sequence lasts for a total of 17 terms, including 7 and 1.

Write a ***recursive*** function that takes as its input a positive integer n and returns the number of values in the sequence generated by n using the procedure described above. You may assume that the sequence terminates in 1000 steps or fewer and that none of the intermediate values generated overflow the int data type.

```
int threeNPlusOne(int n) {

    if ( n == 1 )                          // Grading 1 pts
        return 1 ;                         // Grading 1 pt

    if ( n%2 == 0)

        return 1 + threeNPlusOne(n/2) ;    // Grading 4 pts

    return 1 + threeNPlusOne(3*n+1) ;      // Grading 4 pts
}
```

8) (10 pts) The following function is an attempt at fast modular exponentiation. Much to the chagrin of the student who wrote it however, it seems to run equally slow as the iterative version. Explain why this code runs slower than expected. What needs to be done to speed it up?

```
long long fastModExp(long long base, long long exp, long long mod) {

    if (exp == 0)
        return 1L;

    if (exp%2 == 0)
        return (fastModExp(base,exp/2,mod)* fastModExp(base,exp/2,mod))%mod;

    return (base*fastModExp(base,exp-1,mod))%mod;
}
```

The problem is that in the second if, once we make the recursive call with exp/2, instead of storing that answer somewhere and multiplying it by itself, we REDO ALL of that work by making the identical recursive call AGAIN. We can speed this up by storing the result of the first recursive call in a variable tmp and then returning (tmp*tmp)%mod;

Grading: 6 pts for saying that the reason it's slow is the redundant recursive call, 4 pts for explaining how to fix it.

9) (10 pts) Write a ***recursive*** function that calculates the sum of the digits of its input integer n. (Note: n is guaranteed to be non-negative.)

```
int sumDigits(int n) {

    if (n == 0)                      // Grading 2 pts (can also be
        return 0;                    // Grading 1 pt   n < 10 ret n)

    return n%10 + sumDigits(n/10); // Grading: 1 pt return
                                   //          2 pts n%10
                                   //          1 pt +
                                   //          3 pts rec call


}
```

2016 Spring COP 3502 Exam #1 Solution - Sorting (10 pts)

Date: 2/18/2016

10) (12 pts) Consider a Merge Sort of the 8 elements shown below. In the process of the sort, the Merge function gets called 7 times. Show the contents of the array after each of the Merge function calls completes (except the last):

Index	0	1	2	3	4	5	6	7
Original	13	27	12	9	30	15	6	3
After 1st Merge	**13**	**27**	**12**	**9**	**30**	**15**	**6**	**3**
After 2nd Merge	**13**	**27**	**9**	**12**	**30**	**15**	**6**	**3**
After 3rd Merge	**9**	**12**	**13**	**27**	**30**	**15**	**6**	**3**
After 4th Merge	**9**	**12**	**13**	**27**	**15**	**30**	**6**	**3**
After 5th Merge	**9**	**12**	**13**	**27**	**15**	**30**	**3**	**6**
After 6th Merge	**9**	**12**	**13**	**27**	**3**	**6**	**15**	**30**
After 7th Merge	3	6	9	12	13	15	27	30

Grading: 2 pts per row, give 2 pts if completely correct, 1 pt if 1/2 the numbers or more or right, 0 pts otherwise

11) (6 pts) Show the contents of the following array after each iteration of Bubble Sort.

Index	0	1	2	3	4	5	6	7
Original	13	27	12	9	30	15	6	3
After 1st Iteration	**13**	**12**	**9**	**27**	**15**	**6**	**3**	**30**
After 2nd Iteration	**12**	**9**	**13**	**15**	**6**	**3**	**27**	**30**
After 3rd Iteration	**9**	**12**	**13**	**6**	**3**	**15**	**27**	**30**
After 4th Iteration	**9**	**12**	**6**	**3**	**13**	**15**	**27**	**30**
After 5th Iteration	**9**	**6**	**3**	**12**	**13**	**15**	**27**	**30**
After 6th Iteration	**6**	**3**	**9**	**12**	**13**	**15**	**27**	**30**
After 7th Iteration	3	6	9	12	13	15	27	30

Grading: 1 pt per row, all or nothing

12) (2 pts) The singer Adele Laurie Blue Adkins is better known by what first name?

Adele Grading: Give 2 pts to all