Computer Science Foundation Exam

August 12, 2011

Section I B

COMPUTER SCIENCE

SOLUTION

NO books, notes, or calculators may be used, and you must work entirely on your own.

Question #	Max Pts	Category	Passing	Score
1	10	ALS	7	
2	10	DSN	7	
3	10	DSN	7	
4	10	ALG	7	
5	10	ALG	7	
TOTAL	50			

You must do all 5 problems in this section of the exam.

Problems will be graded based on the completeness of the solution steps and not graded based on the answer alone. Credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

1) (10 pts) ALS (Order Analysis)

For the code segment shown below,
(a) (4 pts) Find the Big-Oh order of this code segment in terms of n.
(b) (6 pts) Determine the final value of x in terms of n.

```
int x = 0;
for (i = 1; i <= (2*n); i++) {
    for (j = 1; j <= n; j++)  {
        if ( j < i)
           x = x + 1;
    }
}
```

(a)

$$\sum_{i=1}^{2n}\sum_{j=1}^{n}1=\sum_{i=1}^{2n}n=n\sum_{i=1}^{2n}1=n(2n)=\mathrm{O}(n^2)$$

Grading: 1 pt for noting the nested structure, 2 pts for noting that each loop runs O(n) times, 1 pt for multiplying the two values together.

(b)

$$\sum_{i=1}^{2n}\sum_{j=1}^{n}1-\sum_{j=1}^{n}j=n\sum_{i=1}^{2n}1-\frac{n\left(n+1\right)}{2}=n(2n)-\frac{n\left(n+1\right)}{2}=2n^2-\frac{n\left(n+1\right)}{2}$$

$$=\frac{4n^2-n^2-n}{2}=\frac{3n^2-n}{2}$$

Grading: 4 pts for figuring that x gets incremented once, twice, etc. at first, for a total of (n-1)n/2 times. Then 2 pts for noticing that x gets incremented n times the last n iterations of the outer loop.

2) (10 pts) DSN (Recursive Algorithms)

Write a recursive function that will multiply an integer m by another integer n (where n > 0) and return the correct result.

```
/*
 * function performs integer multiplication
 * precondition:  m and n are defined and n > 0
 * postcondition: returns m*n
 */
int  multiply(int m, int n) {

    int answer;

    if (n == 1) /* base case */          // 2 pts.
        answer = m                       // 1 pt
    else  /* recursive step */           // 1 pt
        answer = m + multiply(m, n-1);  // m is 1 pt, + is 1 pt
                                         // multiply is 1 pt,
                                         // m is 1 pt, n-1 is 1 pt
    return answer;                       // 1 pt
}
```

3) (10 pts) DSN (Linked Lists)

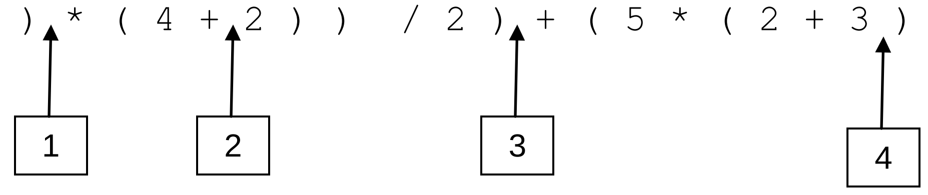
Write a function that will take in a pointer to the front of a linked list, exchange the first node in the list with the second node, and return a pointer to the front of this edited list. If the original list has fewer than two nodes, no changes should be made and the original pointer to the front of the list should be returned. Please use the struct and function prototype provided below:

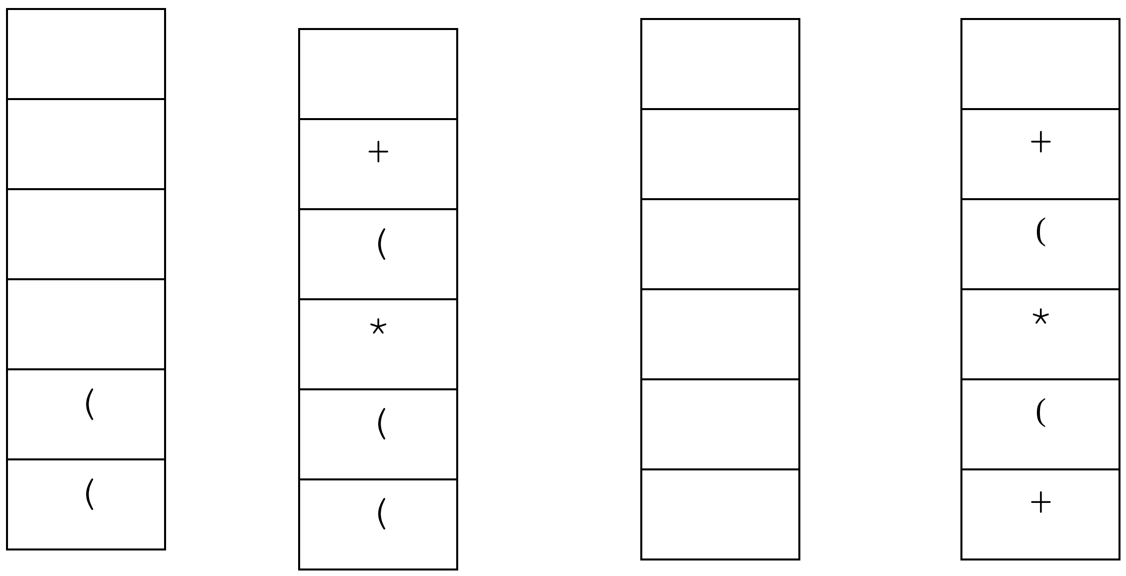
```
struct node {
    int data;
    struct node *next;
};

struct node* swapFirstTwo(struct node* list) {

    // Base cases: 1 pt if, 1 pt NULL, 1 pt 1 node, 1 pt return
    if (list == NULL || list->next == NULL)
        return list;

    // 1 pt – though this step can be avoided
    struct node* newFront = list->next;

    // 2 pts
    list->next = newFront->next;

    // 2 pts
    newFront->next = list;

    // 1 pt
    return newFront;

}
```

4) (10 pts) ALG (Stacks)

Convert the following infix expression to its corresponding postfix form using a stack. Show the values in the stack at the points indicated in the expression (points marked 1, 2, 3, and 4).

```
( ( ( 1 + 3 ) * ( 4 + 2 ) )  / 2 ) + ( 5 * ( 2 + 3 ) )
```





Stack at point 1

Stack at point 2

Stack at point 3

EMPTY

Stack at point 4

Postfix format:

```
1 3 + 4 2 + * 2 / 5 2 3 + * +
```

Grading: 1 pt off for each distinct mistake made. Trace through their work to figure out how many actual mistakes were made. Cap at 10.

5) (10 pts) ALG (Sorting)

(a) (3 pts) The array shown below has just been partitioned once. (Remember that the partition is a function used in Quick Sort.) Determine the partition element and give a justification for your answer.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Value	12	14	10	16	27	23	45	19	18	66	33	25	31

Partition element: **16, or the element in index 3 (1 pt)**

Justification: **Everything in indexes 0 through 2 is less than 16, everything in indexes 4 through 12 is greater than 16. This is true for no other element in the array, but must be true about the partition element of an array. (2 pts)**

(b) (3 pts) Let T(n) represent the run-time of a Merge Sort of n elements. Write down a recurrence relation that T(n) satisfies that is based on the standard recursive implementation of the sort.

Recurrence Relation: **$T(n) = 2T(n/2) + O(n)$, 1 pt for 2, 1 pt for $T(n/2)$ and 1 pt for $O(n)$.**

(c) (4 pts) Show the contents of the following array after each iteration of an insertion sort:

Iteration	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7	Index 8
0	12	3	8	14	7	6	9	2	10
1	**3**	**12**	**8**	**14**	**7**	**6**	**9**	**2**	**10**
2	**3**	**8**	**12**	**14**	**7**	**6**	**9**	**2**	**10**
3	**3**	**8**	**12**	**14**	**7**	**6**	**9**	**2**	**10**
4	**3**	**7**	**8**	**12**	**14**	**6**	**9**	**2**	**10**
5	**3**	**6**	**7**	**8**	**12**	**14**	**9**	**2**	**10**
6	**3**	**6**	**7**	**8**	**9**	**12**	**14**	**2**	**10**
7	**2**	**3**	**6**	**7**	**8**	**9**	**12**	**14**	**10**
8	**2**	**3**	**6**	**7**	**8**	**9**	**10**	**12**	**14**

Grading: ½ point per row that is completely correct, round down.