Computer Science Foundation Exam

May 8, 2015

Section I A

COMPUTER SCIENCE

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

SOLUTION

Question #	Max Pts	Category	Passing	Score
1	10	DSN	7	
2	10	ANL	7	
3	10	ALG	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) DSN (Recursive Functions)

Imagine a 2-D array of non-negative integers where each item represents the number of gold bars in the corresponding region of a map. Assume that you are placed in a particular square and want to collect the gold that is reachable without moving into a square that initially has 0 gold bars. You may move up, down, left or right. Complete the function `getGold` below so that it completes a floodfill of the designated area, filling all reachable squares (from the initial square) with 0, and returns the total number of gold bars collected. Assume that `SIZE` is a positive integer constant and that grid has dimensions `SIZE x SIZE`. An `inbounds` function and constants `DX` and `DY` have been provided for convenience.

```
const int DX[] = {-1,0,0,1};
const int DY[] = {0,-1,1,0};

int getGold(int grid[][SIZE], int row, int col) {

    if (!inbounds(row, col)) return 0;
    if (grid[row][col] == 0) return 0;

    int bars = grid[row][col] , i; // Grading 2 pts - 1 pt grid
                                   // 1 pt both indexes.
    grid[row][col] = 0 ;           // 1 pt

    for (i = 0; i<4; i++)

        bars += getGold(grid, row+DX[i], col+DY[i]);
        // Each component 1 pt - bars, +, =, getGold, grid,
        //                       row+DX[i] and col+DY[i]
    return bars;
}

int inbounds(int row, int col) {
    return row >= 0 && row < SIZE && col >= 0 && col <SIZE;
}
```

2) (10 pts) ANL (Summations and Algorithm Analysis)

Find the closed form solution in terms of n for the following summation. Be sure to show all you work.

$$\sum_{k=5}^{2n}(3k-2)$$

$$\sum_{k=1}^{2n}(3k-2)-\sum_{k=1}^{4}(3k-2)$$
$$\sum_{k=1}^{2n}3k-\sum_{k=1}^{2n}2-\sum_{k=1}^{4}3k+\sum_{k=1}^{4}2$$
$$3\sum_{k=1}^{2n}k-2\sum_{k=1}^{2n}1-3\sum_{k=1}^{4}k+2\sum_{k=1}^{4}1$$
$$3\left[\frac{2n(2n+1)}{2}\right]-2[2n]-3\left[\frac{4*5}{2}\right]+2[4]$$
$$3\left[\frac{4n^2+2n}{2}\right]-4n-30+8$$
$$3[2n^2+n]-4n-22$$
$$6n^2+3n-4n-22$$
$$6n^2-n-22$$

// Grading:
// 2 pts for changing the bounds of the summation
// 1 pt for splitting the terms
// 1 pt for moving the constants outside
// 2 pts for applying the correct summation rule
// 1 pt for simplifying terms after applying summation rule
// 1 pt for reducing equation through division by 2
// 1 pt for multiplying 3 with reduced terms
// 1 pt for final solution

3) (10 pts) ALG (Queues)

For this question, use the struct definition provided below:

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

Also, assume that a struct queue has been defined as well as functions with the following prototypes, that operate according to the names of the functions:

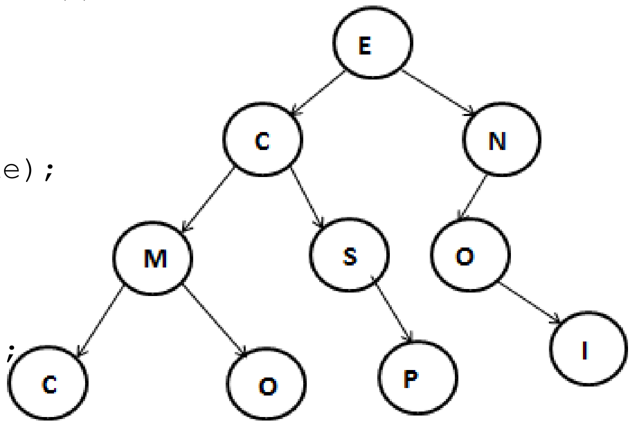
```
void enqueue(queue* myQueue, treenode* node);
void initialize(queue* myQueue);
treenode* dequeue(queue* myQueue);
int empty(queue* myQueue);
```

Show the output of running the function call `printTrace(root)` where printTrace is the function shown below and root is pointing to the root of the tree (storing E) shown below.

```
void printTrace(treenode* root) {

  queue* myQueue = malloc(sizeof(queue));
  initialize(myQueue);
  enqueue(myQueue, root);

  while (!empty(myQueue)) {
    treenode* next = dequeue(myQueue);
    printf("%c", next->letter);
    if (next->left != NULL)
      enqueue(myQueue, next->left);
    if (next->right != NULL)
      enqueue(myQueue, next->right);
  }
}
```



Output:

E	**C**	**N**	**M**	**S**	**O**	**C**	**O**	**P**	**I**

Grading: 1 pt per letter - must be in correct slot.

4) (10 pts) ALG (Binary Heaps)

Recall that a binary heap of n elements is internally stored in an array, from index 1 through index n, inclusive. (Index 0 is typically left blank.) Consider the binary heap represented by the array shown below:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Value		4	17	6	22	45	13	9	47	25	50	51	92

(a) (5 pts) Show the result of inserting the item 15 into the heap. You may sketch out the tree form of the heap below, but copy your answer into the array provided. Only this array will be graded.

Initial heap:

```
               4
          /         \
     17                  6
   /    \             /    \
22        45       13        9
/  \     /  \     /
47  25  50  51   92
```

After insert:

```
               4
          /         \
     17                  6
   /    \             /    \
22        45       13        9
/  \     /  \     /  \
47  25  50  51   92  15
```

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Value		**4**	**17**	**6**	**22**	**45**	**13**	**9**	**47**	**25**	**50**	**51**	**92**	**15**

Grading: 5 pts for having 15 in the correct spot, 0 pts for anything else.

(b) (5 pts) Show the result of deleting the minimum item (4) from the original heap above. You may sketch out the tree form of the heap below, but copy your answer into the array provided. Only this array will be graded.

Right after delete:

```
               92
          /         \
     17                  6
   /    \             /    \
22        45       13        9
/  \     /  \
47  25  50  51
```

After completion of delete:

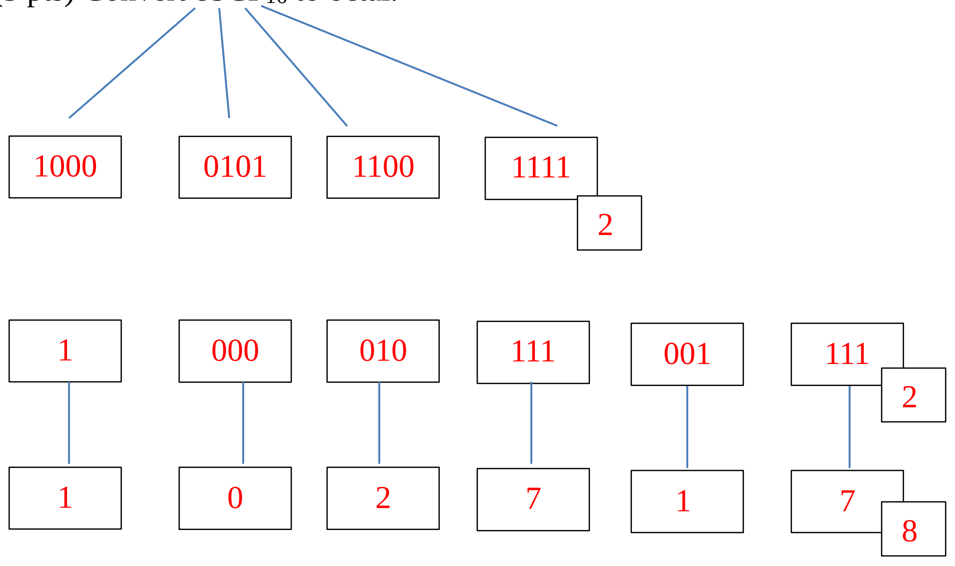
```
               6
          /         \
     17                  9
   /    \             /    \
22        45       13        92
/  \     /  \
47  25  50  51
```

Index	0	1	2	3	4	5	6	7	8	9	10	11
Value		**6**	**17**	**9**	**22**	**45**	**13**	**92**	**47**	**25**	**50**	**51**

Grading: 1 pt for leaving left untouched, 2 pts swap with 6, 2 pts swap with 9.

5) (10 pts) ALG (Base Conversion)

(a) (5 pts) Convert $85CF_{16}$ to octal.



102717_8

Grading: 2 pts for converting from base 16 to base 2, 1 pt for regrouping from sets of four values to sets of three values, 2 pts for converting regrouped base 2 to base 8.

(b) (5 pts) Convert 5815_{10} to hexadecimal.

5815/16	=	363	with remainder	7
363/16	=	22	with remainder	11 (B)
22/16	=	1	with remainder	6
1/16	=	0	with remainder	1

Answer: $5815_{10} = 16B7_{16}$

Grading: 1 pt for each division by 16 with remainder, 1 pt for final solution in correct order