Computer Science Foundation Exam

May 3, 2013

Section I B SOLUTION

COMPUTER SCIENCE

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

Name: _____

PID: _____

Question #	Max Pts	Category	Passing	Score
1	10	ANL	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 <u>not</u> 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 <u>be neat</u>.

1) (10pts) ANL (Algorithm Analysis)

(a) (6 pts) List the best case and worst case of each of the following algorithms/operations in terms of their input size, n:

(i) Inserting an item into a Linked List of n elements in sorted order

best case: O(1) worst case: O(n) (Grading: 1 pt each)

(ii) A Quick Sort of n elements

best case: O(nlgn) worst case: $O(n^2)$ (Grading: 1 pt each)

(iii) Accessing an element in a hash table

best case: O(1) worst case: O(n) (Grading: 1 pt each)

(b)(4 pts) For each of the code snippets below, write the run-time for each snippet of code, using order notation in terms of the variable N:

```
int i = 0, j = 0, k = 0;
for (i = 0; i < N; i++)
   for(j = 0; j < i; j++)
      k += j;
```

(Grading: 1 pt each)

$O(N^2)$

```
int i = 0, j = 0, k = 0;
for (i = 0; i < N; i++)
   for(j = 0; j < 2*N; j++)
      k += j;
```

$O(N^2)$

```
int i = 0, j = 0, k = 0;
for (i = 0; i < N; i++)
   for(j = 0; j < 10; j++)
      k += j;
```

O(N)

```
int i = 0, j = 0, k = 0;
for (i = 0; i < N; i++)
   for(j = N; j > 0; j=j/2)
      k += j;
```

O(NlgN)

---

2) (10 pts) DSN (Recursive Algorithms – Binary Trees)

Write a **recursive** function isBSTRec that determines whether a binary tree is also a binary search tree with all values within a given range. Your function will take in a pointer to the root of a binary tree as well both a minimum and maximum value. Your function should return 1 if and only if the tree pointed to by root is a valid binary search tree with all values in between min and max inclusive. Otherwise, the function should return 0. You may assume that root points to a valid node and is not NULL. Complete the function.

```
struct treeNode {
    int data;
    struct treeNode *left;
    struct treeNode *right;
};

int isBST(struct treeNode* root, int min, int max) {

    if ((root->data < min) || (root->data > max))  // 2 pts
        return 0;                                  // 1 pt

    int ans = 1;
    if(root->left != NULL)                         // 3 pts
        ans = ans && isBSTRec(root->left, min, root->data);
    if(root->right != NULL)                        // 3 pts
        ans = ans && isBSTRec(root->right, root->data, max);
    return ans;                                    // 1 pt
}
```

3) (10 pts) DSN (Linked Lists)

Imagine using a linked list to store a string. In particular, each node of the linked list stores a single character, with the leftmost character stored first in the list. Write a function that converts a standard C string into this list representation. You may **not** assume that *string*.h has been included or include it. The function should return a pointer to the front of the list.

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

struct node* str2lst(char *str) {

  struct node dummy_head;
  struct node *curr_ptr = &dummy_head;
  dummy_head.next = NULL;

  int i = 0;
  while(str[i]!='\0')
  {
     struct node *tmp =malloc(sizeof(struct node));
     tmp->next = NULL;
     tmp->data = str[i];
     curr_ptr->next = tmp;
     curr_ptr = tmp;
     i++;
  }
  return dummy_head.next;
}
```

```
struct node* str2lst(char *str) {    // Grading:
                                     //          4 pts – new node
    int i = 0;                       //          3 pts - attaching
    while (str[i] != '\0') i++;      //          3 pts - ordering
    i--;

    struct node* ans = NULL;
    while (i >= 0) {
        struct node* temp = malloc(sizeof(struct node));
        temp->next = ans;
        temp->data = str[i];
        i--;
        ans = temp;
    }

    return ans;
}
```

4) (10 pts) ALG (Base Conversion)
(a)(5 pts) Convert 2989 in base 10 to hexadecimal (base 16) using the deterministic algorithm taught in class. (Note: An ad hoc method using guessing and checking will NOT be given credit!!!)

161 = 16 16 2 = 256 163 = 4096
2989/256 = 11 = B (1 pt)
173/16 = 10 = A (1 pt)
13 = D (1 pt)

ANSWER:BAD_{16} (2 pts)

(b)(5 pts) Convert $C3D0_{16}$ to base 8. (Note: Use an intermediate base for conversion. Either base 2 or base 10 will be accepted as this intermediate base.)

C3D0 = $1100\ 0011\ 1101\ 0000_2 = 141720_8$

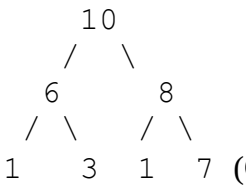
Grading:4 pts for binary, 1 pt to convert to octal

5) (10 pts) ALG (Sorting)

(a) (2 pts)
Is the array `10 6 8 1 3 1 7` a min-heap or a max-heap?

Max-heap (Grading: 0 or 2 pts)

(b) (2pts) Draw the binary tree representation of the heap



(Grading: 0 or 2 pts)

(c) (2 pts) After an element is removed from the heap, write the values in the array, in the order they appear in the array.

8 6 7 1 3 1 (2 pts correct, 1 pt if 4 or more are correct, 0 otherwise)

(d) (4 pts) Write the values in the array after the Partition algorithm from QuickSort is run on the array shown below. Assume that the pivot is 37.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Values	37	98	5	41	25	44	99	79	92	6	2	11	87	45	63

Give full credit to these two solutions: (1 pt off for each incorrect swap)

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Values	11	2	5	6	25	37	99	79	92	41	98	63	87	45	44

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Values	6	11	5	2	25	37	99	79	92	44	41	98	87	45	63

When grading this, there should be generous partial credit if the result is valid (ie, everything to the left of 37 is below 37 and everything to the right is greater), but some values are out of place.