Computer Science I – Exam 2 (REVIEW)

October 21, 2011

Name: ______________________________

PID: ______________________________

	Max Pts	Type	Score
Q1	10	Recurrence Relation	
Q2	10	Infix/Postfix	
Q3	10	Postfix	
Q4	10	Queues	
Q5	12	Code	
Q6	8	BST	
Q7	10	BST	
Q8	8	BST	
Q9	12	Code	
Q10	10	Base Conversion	
Total	100		

Partial credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat.

Do your rough work on the last page.

1) (10pts) Use Use the iteration technique to find the closed form solution of the following recurrence relation. After finding the closed form solution, give the solution in terms of Big-O notation.

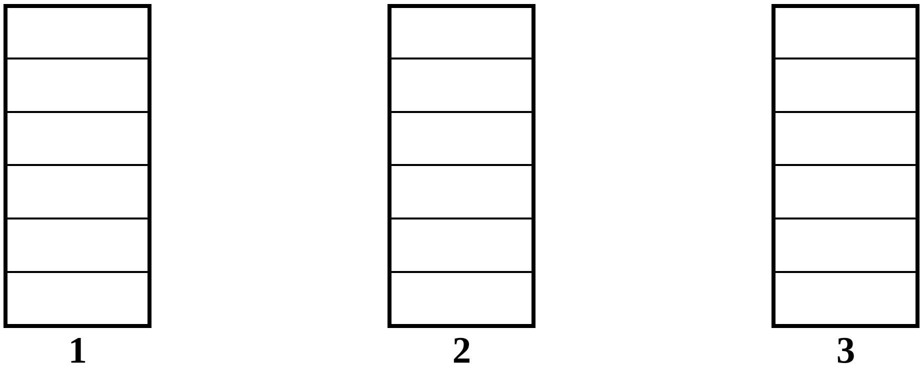
$T(n) = 4T(n/4) + n$ $\qquad$ $T(1) = 4$

Closed Form Solution: ________________________

Big-O Notation: ________________________

2) (10 pts) Convert the following infix expression into its equivalent postfix expression using a stack. Show the contents of the operator stack at the indicated points in the infix expressions (points 1, 2 and 3), and also the final postfix expression. You may draw another stack alongside for your work.

```
                       1                    2               3
A  *  (  B  +  C    /    (  D  +  E )  )  –    F  * ( G  /    H  +  I )
```



postfix expression upto 1 : ____

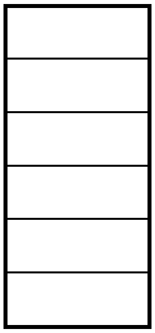
postfix expression upto 2 : ____

postfix expression upto 3 : ____

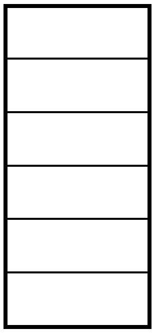
Final postfix expression : ____

3) (8 pts) Stack Applications. Evaluate the following postfix expression using a stack. Additionally, you must show the contents of the stack at the indicated points (1, 2, and 3) in the postfix expression.

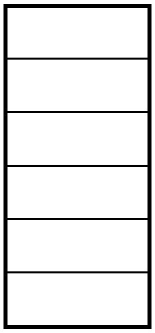
```
        1        2         3
7  14  2  /  2  8  +  4  6  5  +  *  +  –  –
```



1



2



3

4) (15 pts) Assume Q1 and Q2 are queues implemented using the array implementation discussed in class. Show the contents of Q1 and Q2 after the following code is executed, as well as the values for front and numElements.

```
typedef struct {
    int* elements;
    int front;
    int numElements;
    int queueSize;
} queue;

// Assume queueSize = 10 and elements is an array of size 10
// after the function call to init.
queue* Q1 = (queue*)malloc(sizeof(queue));
init(Q1);

queue* Q2 = (queue*)malloc(sizeof(queue));
init(Q2);

// Enqueue/Dequeue some items.
enqueue(Q1, 7);
enqueue(Q1, 8);
enqueue(Q1, 9);
enqueue(Q2, 1);
enqueue(Q2, 2);
enqueue(Q2, dequeue(Q1));
enqueue(Q2, dequeue(Q1));
enqueue(Q1, dequeue(Q2));
```

Q1: elements:

front: ____________
numElements: ____________

Q2: elements:

front: ____________
numElements: ____________

5) Using the operations on stacks (function prototypes given below), write a function that takes in an array of integers and then uses a stack to reverse the array and returns the new reversed array.

```
void initialize(stack* stackPtr);
int full(stack* stackPtr);
int push(stack* stackPtr, int value);
int empty(stack* stackPtr);
int pop(stack* stackPtr);
int top(stack* stackPtr);


int *reverseArray(int *array, int size) {

















}
```

6) (5 pts) Draw a binary search tree created by inserting the following items in this order: 50, 25, 80, 70, 60, 75, 90, 12, 37, 6, 3 and 44.

7) (9 pts) Determine the preorder, inorder and postorder traversals of the following binary tree:

```
                30
            /            \
         10             13
       /     \         /
    67      18       9
   /       /   \        \
14        2   27       34
         /  \           /  \
       12    6        3   81
```

Preorder: ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___

Inorder: ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___

Postorder: ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___

8) (5 pts) What is the result of deleting 37 from the binary tree depicted below?

```
                45
             /         \
          37          87
       /             /     \
    20           56       98
   /   \                   /  \
10    33               92     99
      /  \
   22    36
```

9) Write a function that operates on a binary tree of integers. Your function should sum up the all of the even values in the leaf nodes only. Make use of the function prototype shown below.

Utilize the following struct that stores a binary search tree node for this question:

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

```
int sum_leaf_even(struct treeNode* p)
{









}
```

10) (a)What is the result of converting 365_7 to base 9?

(b)To what does 87 (in decimal) convert in binary?

Scratch Paper: