CSCI 200

HW #1: Insertion sort, analyzing algorithms, growth of functions, basic data structures, hash tables

Directions: *Complete your work on a separate sheet of paper. Submit the physical copy of your work at the beginning of class on the specified due date. Show your work. You may work in groups of up to 3 students provided that all students participate in each question; write all names at the top. Unless otherwise stated, assume that logarithmic functions are using base 2. Provide a short preliminary explanation of how an algorithm works before running an algorithm or presenting a formal algorithm description, and use examples or diagrams if they are needed to make your presentation clear.*

1. Using Figure 2.2 as a model, illustrate the operation of `INSERTION-SORT` on the array $A = [10, 42, 30, 28, 53, 42]$.

2. Consider sorting n numbers stored in an array A, indexed 1 to n, by first finding the smallest element of A and exchanging it with the element in $A[1]$. Then, find the second smallest element of A, and exchange it with $A[2]$. Continue in this manner for the first $n-1$ elements of A (i.e., up to index $n-1$).
   (a) Write pseudocode for this algorithm, which is known as `SELECTION-SORT`
   (b) What loop invariant does this algorithm maintain? Why does it need to run for only the first $n-1$ elements, rather than all n elements?
   (c) What is the best-case and worst-case run time in Θ-notation?

```
function Foo($a$, $n$)
    $k \leftarrow 0$
    $b \leftarrow 1$
    while $k < n$ do
        $k \leftarrow k + 1$
        $b \leftarrow b * a$
    return $b$
```

```
function Bar($a$, $n$)
    $k \leftarrow n$
    $b \leftarrow 1$
    $c \leftarrow a$
    while $k > 0$ do
        if $k \bmod 2 = 0$ then
            $k \leftarrow k/2$
            $c \leftarrow c * c$
        else
            $k \leftarrow k - 1$
            $b \leftarrow b * c$
    return $b$
```

```
function Baz($n$)
    $k \leftarrow 1$
    $m \leftarrow 1$
    while $k \leq 6$ do
        $m \leftarrow m * k$
        $k \leftarrow k + 1$
    return $m$
```

```
function Moo($n$)
    $k \leftarrow 1$
    $x \leftarrow 1$
    while $k \leq n$ do
        $m \leftarrow 1$
        while $m \leq n$ do
            $x \leftarrow m * k$
            $m \leftarrow m + 1$
        $k \leftarrow k + 1$
    return $x$
```

3. (a) Consider the algorithm `Foo`, which takes as input integers a and n. Describe what the algorithm `Foo` does. *Hint: 'It computes ______.' Fill in the blank with a mathematical expression.*
   (b) Analyze the worst-case running time of `Foo` and express it in O-notation. Justify.

4. (a) Consider the algorithm `Bar`, which takes as input integers a and n. Describe what the algorithm `Bar` does. *Hint: 'It computes ______.' Fill in the blank with a mathematical expression.*
   (b) Analyze the worst-case running time of `Bar` and express it in O-notation. Justify.

5. (a) Consider the algorithm `Baz`, which takes as input integer n. Describe what the algorithm `Baz` does. *Hint: 'It computes ______.' Fill in the blank with a mathematical expression.*
   (b) Analyze the worst-case running time of `Baz` and express it in O-notation. Justify.

6. (a) Consider the algorithm `Moo`, which takes as input integer n. Describe what the algorithm `Moo` does. *Hint: 'It computes ______.' Fill in the blank with a mathematical expression.*
   (b) Analyze the worst-case running time of `Moo` and express it in O-notation. Justify.

7. Express each of the following functions using O-notation. That is, find the slowest growing function $g(n)$ such that $f(n) \in O(g(n))$.
   (a) $f(n) = 10n + 32$
   (b) $f(n) = 5\log(n) + 42$
   (c) $f(n) = 18 + n^5 + 2^n$
   (d) $f(n) = 4 - 4n + 4n^2 + 4n^3$
   (e) $f(n) = 5\log(n) + 42n$

8. Rank the following functions by order of growth, from slowest-growing to fastest-growing. That is, find an arrangement $f_1, f_2, f_3, ..., f_{10}$ of the following functions satisfying $f_1 \in O(f_2)$, $f_2 \in O(f_3)$, etc.

$2^{\log_2(n)}$	n^{100}	100	$n^{0.001}$	$log_2(n)$
$\log_2^3(n)$	$n\log_2(n)$	2^n	n^2	$\sqrt{n}$

9. (a) You are given an initially empty stack S stored in array $S[1...10]$. Perform the following operations on it: PUSH(S, 'B'), PUSH(S, 'A'), PUSH(S, 'T'), PUSH(S, 'I'), POP(S), POP(S), PUSH(S, 'Z'), PUSH(S, 'A'), POP(S), PUSH(S, 'I'), PUSH(S, 'N'), PUSH(S, 'L'), POP(S), PUSH(S, 'G'), PUSH(S, 'A'), PUSH(S, 'R'), PUSH(S, 'F'), POP(S), POP(S). Show the contents of the stack after all operations have been performed, using figure 10.1 as a model, and indicate where the top of the stack is.
   (b) You are given an initially empty queue Q stored in array $Q[1...10]$. Perform the following operations on it: ENQUEUE(Q, 'B'), ENQUEUE(Q, 'A'), ENQUEUE(Q, 'T'), ENQUEUE(Q, 'I'), DEQUEUE(Q), DEQUEUE(Q), ENQUEUE(Q, 'Z'), ENQUEUE(Q, 'A'), DEQUEUE(Q), ENQUEUE(Q, 'I'), ENQUEUE(Q, 'N'), ENQUEUE(Q, 'L'), DEQUEUE(Q), ENQUEUE(Q, 'G'), ENQUEUE(Q, 'A'), ENQUEUE(Q, 'R'), ENQUEUE(Q, 'F'), DEQUEUE(Q), DEQUEUE(Q). Show the contents of the queue after all operations have been performed, using figure 10.2 as a model, and indicate where the head and tail of the queue are.

10. Explain how to implement two stacks in one array $A[1...n]$ in such a way that neither stack overflows unless the total number of elements in both stacks together is n. The PUSH and POP operations for both stacks should run in $O(1)$ time.

11. Give the **pseudocode** for a linear-time algorithm for reversing a queue Q. To access the queue, you are only allowed to use the methods of a queue abstract data type (ENQUEUE, DEQUEUE, QUEUE-EMPTY). *Hint: You may use an auxiliary data structure.*

12. Explain how to implement a queue using a singly linked list L. The operations ENQUEUE and DEQUEUE should still take $O(1)$ time.

13. Draw a single binary tree T such that all of the following properties hold for T:
   - each node of T stores a single character,
   - a preorder traversal of T yields COMPILE, and
   - an inorder traversal of T yields PMIOLCE

14. Give the **pseudocode** for an algorithm called COMPUTEDEPTH(T) which runs in total $O(n)$-time and computes the depth of every node of a tree T, where n is the number of nodes of T. Assume every node has an attribute `depth` which has an initially null value, i.e., the purpose of the COMPUTEDEPTH algorithm is to appropriately set the depth attribute of every node with its correct depth value. *Hint: After you have set a particular node's depth attribute, you may access it later to help set another node's depth attribute.*

15. Use the table below to convert a character key to an integer. For each of the following questions, give the contents of the hash table that results when the following keys are inserted into an initially empty 13-item hash table, which has indices 0 to 12. The keys to insert, in this order, are: $(E_1, A, S_1, Y, Q, U, E_2, S_2, T, I, O, N)$. Use $h(k) = k \mod 13$ for the hash function for the k-th letter of the alphabet (see above table for converting letter keys to integer values). Use the provided method to resolve collisions. Note that the elements to insert into the hashtable are letters. not numbers.

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M
Key	0	1	2	3	4	5	6	7	8	9	10	11	12
Letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Key	13	14	15	16	17	18	19	20	21	22	23	24	25

(a) Illustrate the contents of the hashtable when collisions are resolved using **chaining.**

(b) Illustrate the contents of the hashtable when collisions are resolved using **linear probing.**

(c) Illustrate the contents of the hashtable when collisions are resolved using **double hashing**. Let the secondary hash function be $h'(k) = 1 + (k \mod 11)$.