UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

FACULTY *of* SCIENCE *and* ENGINEERING

Department of Computer Science
and Information Systems

Final Assessment Paper

Academic Year:	2011/2012	Semester:	Spring
Module Title:	Data Structures and Algorithms	Module Code:	CS4115
Duration of Exam:	$2\frac{1}{2}$ hours	Percent of Semester Marks:	60
Lecturer:	P. Healy	Paper marked out of:	100

Instructions to Candidates:

- There are two sections to the paper: Multiple Choice Questions and Long Questions
- The mark distribution is 40 marks for Multiple Choice Questions and 60 marks for the Long Questions
- Answer all questions in all sections
- Calculators are *not* allowed

Section 1. Multiple Choice Answers (40 marks).

Use the machine-readable multiple-choice question grid that has been provided to answer these questions. Please completely mark in black exactly one circle on the grid for each answer. A penalty will be charged for wrong answers. Mark the **X** bubble for those questions you wish to skip.

1. Which of the answers below is $\sum_{i=0}^{\infty} 3^{-i}$?
   (a) $\frac{2}{3}$
   (b) ∞
   (c) $\frac{3}{2}$
   (d) None of the above

2. Which of the answers below is $\sum_{i=2}^{\infty} 3^{-i}$?
   (a) $\frac{2}{3} \times \frac{1}{3^2}$
   (b) $\frac{1}{6}$
   (c) ∞
   (d) None of the above

3. Which of the answers below is $\sum_{i=2}^{n+1} 6^i$?
   (a) $\frac{36}{5}(6^n - 1)$
   (b) $\frac{1}{5}(6^{n-1} - 1)$
   (c) $\frac{1}{5}(6^n - 1)$
   (d) None of the above

4. Which of the answers below is $\log_b x$?
   (a) $\log_2 b \times \log_2 x$
   (b) $\frac{\log_2 x}{\log_2 b}$
   (c) $\log_x 2$
   (d) $\log_x b$

5. Which of the answers below is $\sum_{i=1}^{50}(2i-1)$?
   (a) 50^2
   (b) 51×50
   (c) 51×52
   (d) 49^2

6. Which of the answers below best approximates $\sum_{i=0}^{n} i^5$?
   (a) $O(i^5)$
   (b) $O(n^4)$
   (c) $O(i^6)$
   (d) None of the above

7. Which of the following best approximates $\sum_{i=1}^{\frac{n}{2}} \frac{1}{i}$?
   (a) $\frac{2}{n}$
   (b) $\frac{1}{n^2}$
   (c) $\log_e n$
   (d) $\log e$

8. Running-time $O(m+n)$ is equivalent to

   A $O(m) + O(n)$

   B $O(\max(m,n))$

   Which of these possiblities are correct?
   (a) Both **A** *and* **B**
   (b) Only **A**
   (c) Only **B**
   (d) Neither **A** *nor* **B**

9. What is the running-time in Big-Oh notation of the following chunk of code?

```
for (int i = 0; i < n; i++)
  i = n-i;
```

   (a) $O(n)$
   (b) $O(1)$
   (c) $O(n^2)$
   (d) $O(n \log n)$

10. Let $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$. Given statements

    A $\frac{T_1(n)}{T_2(n)} = O(1)$;

    B $T_1(n) + T_2(n) = O(f(n))$

    which of them are true?
    (a) Both **A** *and* **B**
    (b) Only **A**
    (c) Only **B**
    (d) Neither **A** *nor* **B**

11. The time-complexity of searching for an item amongst an array of *sorted* items is (most precisely)
    (a) $\Theta(n \log n)$
    (b) $o(\log n)$
    (c) $O(\log n)$
    (d) $\Omega(\log n)$

12. What is the time-complexity of the following piece of code in "Big-Oh" notation?

```
sum = 0;
for (int i = 0; i < n; i++)
  for (j = 1; j < n; j = j*2)
    sum += n;
```

    (a) $O(n^2)$
    (b) $O(n)$
    (c) $O(\log n)$
    (d) $O(n \log n)$

13. Which is larger, 18^9 or 9^{18}?
    (a) 18^9
    (b) 9^{18}
    (c) They are both the same
    (d) In the limit they will differ only slightly

14. How many nodes are contained in a perfect ternary (3-way) tree of height k (levels $0 \ldots k$)?
    (a) $\frac{3^k}{2}$
    (b) $\frac{k^3-1}{2}$
    (c) $\frac{3^{k+1}-1}{2}$
    (d) 3^3

15. How many nodes are contained in the lowest layer of a perfect ternary (3-way) tree of height k (levels $0 \ldots k$)?

    (a) $\frac{3^k}{2}$

    (b) 3^k

    (c) $\frac{k^3-1}{2}$

    (d) $\frac{3^{k+1}-1}{2}$

16. How many nodes are contained in layers 10 to 23 of a perfect ternary (3-way) tree of height 30 (levels $0 \ldots 30$)?

    (a) All three of the following answers

    (b) $\frac{3^{24}-1}{2} - \frac{3^{10}-1}{2}$

    (c) $\frac{3^{24}-3^{10}}{2}$

    (d) $\frac{3^{10}(3^{14}-1)}{2}$

In the following three questions `BST<Comparable>` is a Binary Search Tree `C++` class into which you may insert "objects" that are comparable, that is, it is possible to decide amongst two which is larger and smaller. Each node of this tree will be a class, `BNode`, with a pointers to the left and right subtrees, `->left` and `->right`, respectively.

17. What does the function `what()` below, (or, more accurately, the member function `BST<Comparable>::what()`), perform?

```
template <class Comparable>
BNode<Comparable> *
BST<Comparable>::what( BNode<Comparable> *t )
{
  if( t == NULL )
    return NULL;
  if( t->left == NULL )
    return t;
  else
    return what( t->left );
}
```

    (a) Count the number of `NULL` nodes in the tree rooted at `t`

    (b) Find the leftmost node in the tree rooted at `t`

    (c) Find a node smaller than `t`

    (d) Find a node larger than `t`

18. What does the (member) function `whatEver()` below, perform? The function `max()` returns the larger of two numbers.

```
template <class Comparable>
int
BST<Comparable>::whatEver( BNode<Comparable> *t )
{
  if( t == NULL )
    return 0;

  int l = whatEver(t->left);
  int r = whatEver(t->right);
  int m = max(l, r); // larger of two ints
  return m+1;
}
```

    (a) Count the number of `NULL` nodes in the tree rooted at `t`

    (b) Count the number of non-`NULL` nodes in the tree rooted at `t`

    (c) Compute the height of the tree rooted at `t`

    (d) Compute the *age* of the tree rooted at `t`

19. What does the (member) function `whatNow()` below, perform?

```
template <class Comparable>
int
BST<Comparable>::whatNow( BNode<Comparable> *t )
{
  if( t == NULL )
    return 0;

  return whatNow(t->left) +
         whatNow(t->right) +
         1;
}
```

    (a) Count the number of `NULL` nodes in the tree rooted at `t`

    (b) Count the number of non-`NULL` nodes in the tree rooted at `t`

    (c) Compute the height of the tree rooted at `t`

    (d) Compute the *age* of the tree rooted at `t`

20. The Fibonacci numbers obey the recurrence relation

$$F_n = F_{n-1} + F_{n-2}, \quad n \geq 2, \quad F_0 = F_1 = 1$$

Which one of the following statements is **false**?

(a) $F_{n+1} = F_n + F_{n-1}, \quad n \geq 1$

(b) $F_{n+2} = F_{n+1} + F_n, \quad n \geq 0$

(c) $F_n = 1 + \sum_{i=0}^{n-2} F_i, \quad n \geq 2$

(d) $F_n = n, \quad n \geq 0$

Section 2. Long Questions (60 marks).

- Please put your answers to these questions in the answer book provided to you
- Label your answers 3.1, 3.2, and 3.3 in your answer books

1. **(20 marks.)**

   (a) Fibonacci numbers have lots of uses in computer science – heaps and trees, for example. Prove by induction that

   $$F_n = 1 + \sum_{i=0}^{n-2} F_i, \quad n \geq 2 \text{ and } \quad F_0 = 0, F_1 = 1$$

   (10 marks.)

   (b) In a binary tree every node has two pointers that are used to point to its two possible offspring. If the node does not have an offspring the unused pointer is called a *null* pointer. Prove that the number of *null* pointers in a binary tree containing n nodes is $n + 1$. (10 marks.)

2. Sorting algorithms. **(20 marks.)**

   (a) Name two possible implementations of Priority Queue abstract data type. That is, name two data structures we have looked at, that could support the PQ ADT. Discuss the advantages and disadvantages of each. (6 marks.)

   (b) Given `arr`, an array of `n` `doubles`, using a Priority Queue, give an algorithm for how the array could be sorted **largest to smallest**. Your algorithm *must* use a Priority Queue. (6 marks.)

   (c) Suppose you have a Binary Search Tree (BST) class in your favourite programming language. In addition to all of the operations you would expect a BST class to support, the class also supports a member function (method) called `inOrderPrint()` that performs an inorder traversal of the tree and prints out the data stored in each node of the tree.

   Based on this class give an algorithm (either in `C++` or at a high level) to print an array of `doubles` in sorted order. (8 marks.)

3. **(20 marks.)**

   A team championship is decided each year by putting all of the teams' names into a hat – there are n of them. The first two names drawn out of the hat play each other; so do the next two names drawn, and so on until there are no names left in the hat. These matches make up Round 1 of the championship.

The losers of these matches are eliminated, the winners go into the hat again and the process repeats, round after round until there is just one overall champion.

In all of the following questions please give *exact* answers, or as exact as the information given allows you to conclude. No Big-Oh, please.

(a) If n is an exact power of 2 (6 marks.)

- over the entire championship, how many games will be played?
- how many rounds will be needed to determine the overall winner?
- how many matches will the champion play?

(b) It would be more usual for the number of teams in the tournament *not* to be an exact power of 2 and might even be an odd number. Answer the three questions of the previous part for general values of n. (6 marks.)

(c) In some competitions the "stronger" teams may not be required to play in early rounds. Suppose that the n teams are separated into n_1 "weak" teams, who have to play from the beginning and n_2 "strong" teams who are allowed enter the competition at Round 3. Answer the same three questions in this case. Make no assumptions about the oddness/evenness of n_1, n_2. (8 marks.)