MONASH University

Office Use Only

Monash University

Semester One Examination Period 2014

Faculty of Information Technology

EXAM CODE: **FIT2004**
TITLE OF PAPER: **Algorithms and Data Structures**
EXAM DURATION: 3 hours writing time
READING TIME: 10 minutes

THIS PAPER IS FOR STUDENTS STUDYING AT:(tick where applicable)

☐ Berwick ✓ Clayton ✓ Malaysia ☐ Off Campus Learning ☐ Open Learning
☐ Caulfield ☐ Gippsland ☐ Peninsula ☐ Enhancement Studies ☐ Sth Africa
☐ Pharmacy ☐ Other (specify)

During an exam, you must not have in your possession, a book, notes, paper, electronic device/s, calculator, pencil case, mobile phone or other material/item which has not been authorised for the exam or specifically permitted as noted below. Any material or item on your desk, chair or person will be deemed to be in your possession. You are reminded that possession of unauthorised materials in an exam is a discipline offence under Monash Statute 4.1.

No examination papers are to be removed from the room.

AUTHORISED MATERIALS

CALCULATORS	☐ **YES**	✓ **NO**
OPEN BOOK	☐ **YES**	✓ **NO**
SPECIFICALLY PERMITTED ITEMS	☐ **YES**	✓ **NO**

Candidates must complete this section if required to write answers within this paper

STUDENT ID __ __ __ __ __ __ __ __ __ DESK NUMBER __ __ __ __

Section		Marks
1		16
2		8
3		10
4		7

Section		Marks
5		12
6		8
7		9
Total		70

Office use only

INSTRUCTIONS

- You must answer ALL the questions.
- Answers to each question should be in the space DIRECTLY BELOW the questions and (if required) on the blank page overleaf of each question.
- Script book may be used if ADDITIONAL SPACE is required for answering these questions

General exam technique

Some candidates throw marks away by not attempting all questions. Suppose you get 7/10 on a question for a 20 minutes effort. Spending another half hour on the question gets *at most* 3 more marks. On the otherhand, if you spend that time on a new question, you might get another 10 marks, or more.

Answer the question that is asked. If the question asks for Insertion sort, do not give Quicksort. Where necessary, especially where it says "No explanation, no marks", justify your answer with a clear explanation.

Some of the questions ask you to write Pseudocode. A Pseudocode is essentially a high-level description of your program, that should allow a human to understand it. If you feel more comfortable using Python/Java syntax, you are welcome to use it but don't get bogged down in syntax. What will essentially be assessed in such questions is your basic understanding of the algorithm (and not the syntactical correctness).

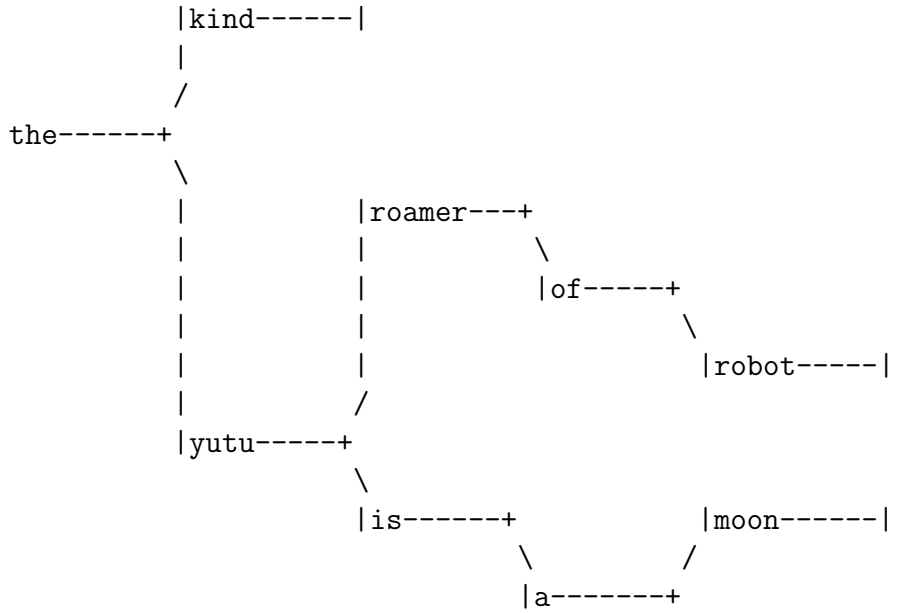
Write neatly. Use a new pen not an old, splodgy one.

Good Luck!

Section 1 (**Short Questions**) **(Section weight = 16 marks)**

This section is composed of 7 questions: (A) to (F)

(A) Consider the following binary tree of words (**depicted in an anticlockwise-rotated sense** with the **root node on the left** and **leaves on the right**):



Write the string of words formed by recursively traversing this tree over each of the following traversals:

(i) Preorder/prefix traversal.

(ii) Postorder/postfix traversal.

(iii) Inorder/infix traversal.

(iv) Breadth-First traversal.

(2 marks)

2

(B) The amount of time, T_N taken by an algorithm is typically a function of the size, N, of the input data. Assume that you wrote a program that shows the following time recurrence:

$$T_N = \begin{cases} \phi T_{N-1} + a, & \text{if } N > 0 \\ b & \text{if } N = 0, \end{cases}$$

where ϕ, a and b are constants.

(i) Work out the solution of T_N from this recurrence.

(ii) What is the time complexity of your program in Big-O notation?

(3 marks)

(C) Assume you are given an array of N integers, `a[1...N]`.

(i) Write a pseudocode to sort these numbers using insertion sort. (Don't write any function and arguments to the function in your pseudocode. Treat the array as given and just focus on writing down as pseudocode, the core essence of insertion sort of this array.)

(ii) Identify the loop invariant of your algorithm.

(iii) In Big-O notation, what is the best-case time-complexity and worst-case space-complexity of insertion sort?

(3 marks)

3

(D) An array `arr[1...N]` contains N characters from a binary alphabet. Write an algorithm (pseudocode) that sorts these characters in $O(N)$-time and $O(1)$-space.

(3 marks)

3

(E) Provide the dynamic programming recurrence relationships (including boundary conditions) for the problem of finding the edit distance between two strings `s1[1...M]` and `s2[1...N]`. **(2 marks)**

2

(F) Assume that you have already implemented correctly a Heap data structure with `upHeap(argument)` and `downHeap(argument)` operations, as discussed in the unit. You are now given an array of distinct random numbers `arr[1...N]`.

Write the **most efficient** code snippet you can to **Heapify** (or convert into a heap) this given array (while treating the above operations as being already a part of your code). Note, your pseudocode should not be more than a loop and its body. Justify your answer.

(3 marks)

3

Section 2 (Abstract Data Types and formal proofs) (Section weight = 8 marks)

This section is composed of 2 questions: (A) and (B)

(A) Given below is the list abstract data type:

```
type List e = nil | cons(e, (List e))
```

An operation `append` and `length` on the list type is defined as follows:

```
append(nil, L2) = L2
| append(L1,L2) = append( cons(h1,T1), L2 )
                 = cons( h1, append(T1,L2) )

length(nil) = 0
| length(L) = length( cons(h,T) ) = 1 + length(T)
```

Formally prove:

```
length(append(L1,L2)) = length(L1) + length(L2)
```

(4 marks)

4

(Page intentionally left blank for working space)

(B) Given below is the abstract data type definition for a binary tree:

```
type tree e = nilTree | fork( e, (tree e), (tree e) )
```

Further, below are definitions of two algorithms on this data structure:

`X` which is an algorithm applicable to trees of any kind, is defined as:

```
X(nilTree)  = nilTree
|  X( fork(e,L,R) ) = fork(e, X(R), X(L))
```

`Y` which is an algorithm applicable only to trees of real numbers:

```
Y(nilTree) = 0
|  Y( fork(e,L,R) ) = e +  Y(L) + Y(R)
```

Prove **formally** that:

(i) X(X(T)) = T, for any tree T.

(ii) Y(X(T)) = Y(T), for every tree T of numbers.

(4 marks)

4

(Page intentionally left blank for working space)

Section 3 (Correctness, Verification and Termination) (Section weight = 10 marks)

This section is composed of 3 questions, (A) to (C)

Background: The pseudocode (program) below gives the core functionality of binary search of `key` in an array of N integers `a[1...N]`

```
1  variable lo=1, hi=N+1;
2  while (lo < hi-1) {
3     variable mid = math_floor((lo+hi)/2);
4
5     if (key >= a[mid])
6        lo=mid;
7     else
8        hi=mid;
9  }
10 if (N > 0 and a[lo] == key) print key_found at lo;
11 else print key_not_found;
```

The next three questions in this section are based on this given program.

(A) (i) What is the precondition that has to be met for this program to work?
(ii) What is the loop invariant of this program?

(2 marks)

2

(B) Explain precisely why the above program is correct. **(5 marks)**

5

(C) Explain precisely why the above program always terminates. **(3 marks)**

3

Section 4 (Search and Retrieval Trees) **(Section weight = 7 marks)**

This section is composed of 2 questions: (A) and (B)

(A) Starting with an initially empty AVL-tree, show what happens as the following elements are inserted in the given order: 50, 70, 30, 10, 20, 15

(4 marks)

4

(Page intentionally left blank for working space)

(B) Given a string of size N, explain in detail why the space-complexity to store a suffix tree (that is, a compact suffix trie) is bounded by $O(N)$. **(3 marks)**

3

Section 5 (Graphs) **(Section weight = 12 marks)**

This section is composed of 3 questions, (A) to (C)

(A) (i) What is the time-complexity of Dijkstra's algorithm when the set of 'Remaining' vertices (as described in this unit) is implemented using a min-heap. Explain your reasoning clearly.

(ii) Explain why Dijkstra's algorithm does not work for graphs with negative weights?

(4 marks)

4

(B) Draw the Directed Graph that corresponds to the following information:

```
Gough is older than Mel, Paul and Kylie.
Claudia is older than Gough and Kylie.
Kylie is older than Natalie.
Natalie is older than Paul.
Mel is older than Natalie and Paul.
```

In addition, suggest if the graph you have drawn is acyclic or not. **(3 marks)**

3

(C) Write a pseudocode to find and print a topological sort, given a Directed Acyclic Graph (DAG) represented by an adjacancy matrix. (Note: Do not write any pseudocode for reading the DAG from an input file; consider the adjacency matrix as given and just write the pseudocode for the algorithm that finds the topological sort and prints it.)

(5 marks)

5

Section 6 (Pattern Matching) **(Section weight = 8 marks)**

This section is composed of 1 (two-part) question

(A) The Burrows-Wheeler Transform (BWT) of a reference string is `AGTTTTCC$GGC`.

(i) Without reconstructing the first column, use LF-mapping to reconstruct **only the last 3 letters** of the original string, excluding the terminal character '`$`'.

(ii) If a given pattern is `ATC`, using **backwards search** described in the unit, how many times does **each suffix** of this pattern occur in the original text?

(The progression of your worked out steps of LF-mapping (in the first part) and backwards search (in the second part) should be clearly written down for your answer to be marked.)

(8 marks)

8

(Page intentionally left blank for working space)

Section 7 (**Miscellaneous**) **(Section weight = 9 marks)**

This section is composed of 3 questions: (A) to (C)

(A) Write the pseudocode for a linear **recursive** program to print the Nth Fibonacci number, where the Fibonacci sequence is $1, 1, 2, 3, 5, 8, 13, \cdots$.

(4 marks)

4

(B) Given is a function $f(x) = x^2 - 2$. With the $x_0 = 1$ as the initial guess what are the values of x_1 and x_2 using Newton's numerical algorithm for finding the root of this function. (Write down clearly the steps involved in arriving at the values x_1 and x_2.)

(2 marks)

2

(C) The *longest common subsequence* (LCS) problem requires one to find the *longest* possible *subsequence* shared between two different strings $s1$ and $s2$. Note that a **subsequence** is *different* from a **substring**, in that the letters in a subsequence need not appear consecutively in the original string, whereas in a substring they do appear consecutively.

For example, if a string s = `abcdefghijklm`, a **subsequence** is some (or all) of its elements, in the same order, for instance, `cfijm` is the subsequence of s. If $s1$ = `arun` and $s2$ = `arvind`, the LCS between these two string is `arn` – there is no other subsequence common between $s1$ and $s2$ that is longer than the common subsequence `arn`.

The LCS problem between any two given strings $s1$ and $s2$ can be solved using a dynamic programming approach, which is similar to the approach used to solving the edit distance problem.

Write the dynamic programming recurrence relationship for this problem, including the boundary conditions.

(3 marks)

3

(Page intentionally left blank for working space)

(Page intentionally left blank for working space)