Office Use Only

Monash University

Faculty of Information Technology

Sample Exam

EXAM CODES: **FIT2004**

TITLE OF PAPER: Data Structures and Algorithms – Final Exam

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**
if yes, items permitted are:		

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

STUDENT ID _ _ _ _ _ _ _ _ _ DESK NUMBER _ _ _ _

Answer all questions in this examination booklet.

1. All answers must be given in the exam booklet. If possible write your answer in the space directly below the question. You may use the backs of the pages to develop your answers, but these pages will not be marked unless you follow the instructions below.
2. If there is insufficient space to complete the answer in the space provided, you can write your answer on the back of the previous page. If you do this you must clearly mark the answer with ANSWER TO QUESTION X, where X is the question number and write ANSWER ON OPPOSITE PAGE in the space provided for the answer.
3. If there is insufficient space on the back of the previous page you may write the answer on the back of another page, but must clearly mark the answer, as above, and must clearly indicate the location of the answer in the space provided for the answer.
4. You must tick the boxes in the table below on this page for all questions that you have attempted.
5. Unless stated otherwise, you only need to give procedural pseudo-code for algorithms. Concrete program code is only required where this is explicitly stated.
6. Where the question asks for pseudocode, you are allowed to use concrete program code if you wish, but this is more difficult and not recommended.

Good Luck!

I declare that I have attempted the questions marked in the table below:

Question	Attempted	Marks (office use only)
1		
2		
3		
4		
5		

Question 1 [Short Questions] [21 marks]

A. Suppose you are considering an application that requires a binary search tree with three operations: 1) to insert a new key, 2) to delete a key and 3) to access a key. What the relative worst case complexities of these three operations if the binary search tree is implemented using respectively an AVL tree or a Splay Tree?

[6 x 0.5 = 3 marks]

	Insert	Delete	Access
AVL Tree			
Splay Tree			

B. What is the advantage of Mergesort over Quicksort when considering runtime complexity? [2 marks]

C. Briefly describe AVL Trees and Red-Black Trees. What are their relative strengths and weaknesses? [4 marks]

D. Describe the Divide and Conquer programming strategy. How does it work? For what types of problems is it useful? [3 marks]

E. Draw a valid heap containing the key elements {7,12,1,3,22,5,11}. Draw your heap as a tree, not as an array [2 marks]

F. What is Polynomial Time? [2 marks]

G. Use proof by induction to prove the following for any $n \geq 0$. [2 marks]

$$\sum_{i=0}^{n} 2i = 2\sum_{i=0}^{n} i$$

Question 2 [Run-time Analysis] [16 marks]

A. [Big-O notation] Explain the meaning of the Big-O notation for asymptotic complexity analysis. **[2 marks]**

B. [Amortized Analysis]

An ordered stack S is a stack where the elements appear in increasing order. It supports the following operations:

- **init**(**S**): Create an empty ordered stack.
- **pop**(**S**): Delete and return the top element from the ordered stack.
- **push**(**S**, **x**): First insert **x** at top of the ordered stack. Then delete all elements larger than **x** to re-establish the increasing order by repeatedly removing the element immediately below **x** until **x** is the largest element on the stack.
- **destroy**(**S**): Delete all elements on the ordered stack. Note, this operation must process every element on the stack.

Like a normal stack we implement an ordered stack as a double linked list.

Using the potential method, Mary would like to determine the amortized running time of each operation in any sequence of *m* **init**, **pop**, **push** or **destroy** operations. She has decided to use the number of tasks in the queue as the potential function. Fill out the following table for Mary. **[6 marks]**

Operation	Actual cost	ΔΦ	Amortized cost
init			
pop			
push			
destroy			

C. [Runtime complexity] What is the runtime complexity of each of the following Java methods? **[2 x 2 = 4 marks]**

i) This method finds the value of the pair of different values whose sum is greatest from an array of positive integer values.

```
static int findPair2(int values[])
{int    best1, best2;

    if (values.length < 2) return 0;

    best1 = values[0];

    for (int i = 1; i < values.length; i++)
    {
        if (values[i] > best1) best1 = values[i];
    }

    best2 = -best1;

    for (int j = 1; j < values.length; j++)
    {
        if (values[j] > best2 && values[j] != best1) best2 = values[j];
    }

    return best1 + best2;
}
```

ANSWER: runtime complexity =

ii) This divide and conquer method finds the n+1th lowest value in an array. It is similar to quicksort, but only explores either the lower or higher values at each level during the recursion.

```
static int quickselect(int values[], int lower, int upper, int n)
{
    if (lower == upper-1) return values[lower];

    int pivot = values[lower];
    int l = lower;
    int u = upper;

    while (l < u) {
        if (values[l] <= pivot) l++;
        else swap(values, l, --u);
    }

    swap(values, lower, --l);

    if (l == n) return values[l];
    else if (l > n) return quickselect(values, lower, l, n);
    else return quickselect(values, u, upper, n);
}
```

ANSWER: runtime complexity =

D. [Runtime recurrence relations] Solve the following recurrence relation explicitly *via substitution* (also known as *telescoping*). **[2 marks]**

$T(1) = 1$
$T(N) = T(N/2) + N$

E. [Runtime recurrence relations] Solve the following recurrence relation explicitly using the *Divide and Conquer Master Equation.* **[2 marks]**

The Divide and Conquer Master Equation for recurrence relations of the form $T(N) = aT(N/b)+cN^k$ is

$$T(N) = \begin{cases} O(N^{\log_b a}) & \text{if } a > b^k \\ O(N^k \log N) & \text{if } a = b^k \\ O(N^k) & \text{if } a < b^k \end{cases}$$

Recurrence relation to solve:

$T(1) = 1$
$T(N) = T(N/2) + N \qquad N > 1$

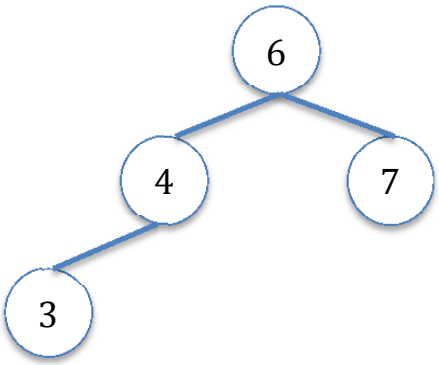
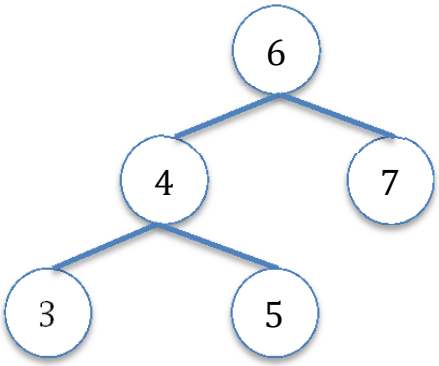
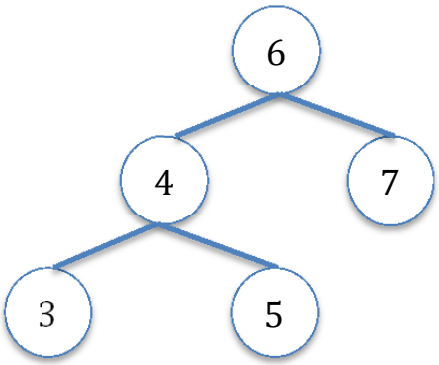
ANSWER:

a **=** _____ ***b*** **=** _____ ***c*** **=** _____ ***k*** **=** _____

T(*N*) =

Question 3 [Tree Data Structures] [20 marks]

A. [Splay Trees] For each of the following Splay Trees and operations on those trees, what does the tree look like after the operation? **[3 * 2 = 6 marks]**

	Tree	Operation	Result
i)		insert(1)	
ii)		search(1)	
iii)		delete(6)	

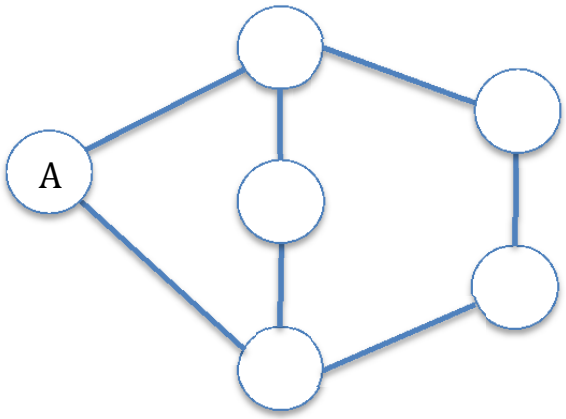
B. [AVL Trees] Show the sequence of trees that results from inserting the values 6, 2, 4, 3, 1 into an initially empty AVL tree. Show the tree after each insertion. **[5 marks]**

C. [Red-Black Trees] Show the sequence of trees that results from inserting the values 6, 2, 4, 3, 1 into an initially empty Red-Black tree. Show the tree after each insertion. Indicate a black node by a box and a red node by a circle. **[5 marks]**

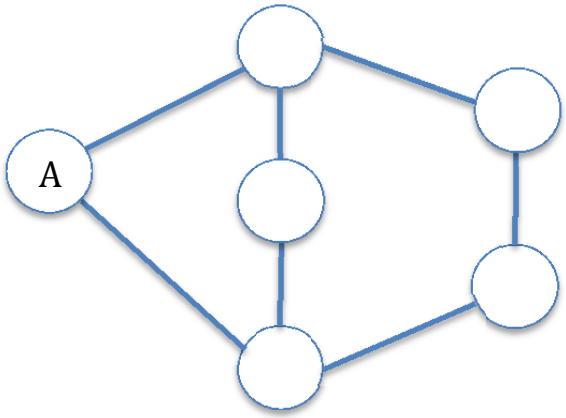
D. [Tries] Draw a compressed suffix trie for the text “ananana”. **[4 marks]**

Question 4 [Graphs and Graph Algorithms] [29 marks]

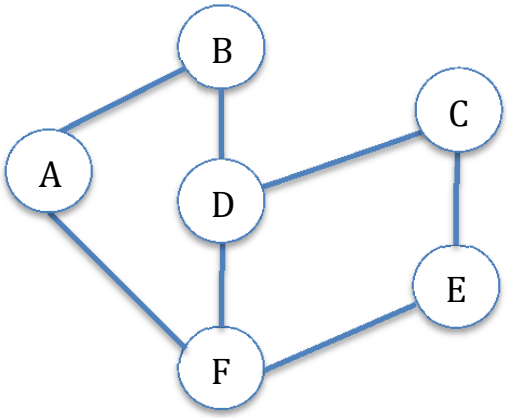
A. Perform a **depth first search** (DFS) on the graph given below starting from the vertex "A". Where a choice exists, use the vertex that is closer to the top of the page. Indicate your answer by numbering the vertices in the order in which they are visited during DFS. **[2 marks]**



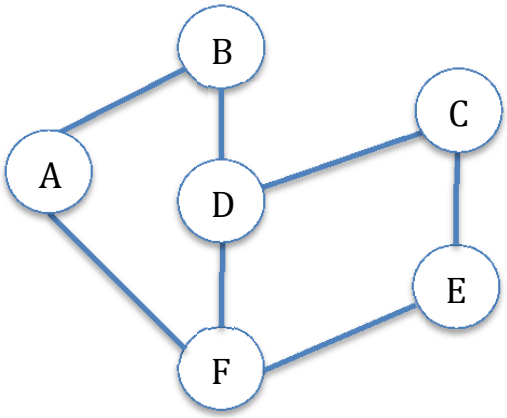
B. Perform a **breadth first search** (DFS) on the graph given below starting from the vertex "A". Where a choice exists, use the vertex that is closer to the top of the page. Indicate your answer by numbering the vertices in the order in which they are visited during DFS. **[2 marks]**



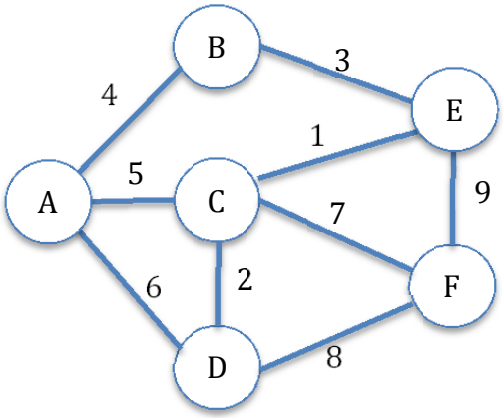
A. Give a visual depiction of the following graph using an **adjacency matrix**. **[1 mark]**



B. Give a visual depiction of the following graph using an **adjacency list**. **[1 mark]**

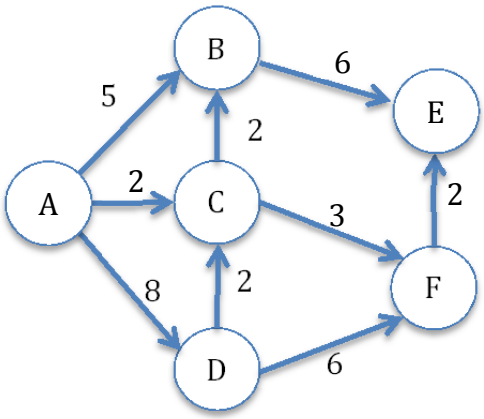


C. Suppose we want to find the minimum spanning tree of the graph below using **Kruskal's algorithm**. List the edges selected by Kruskal's algorithm in the order in which they are selected. **[4 marks]**



D. Draw the *minimum spanning tree* which is found by Kruskal's algorithm in the previous question. **[1 mark]**

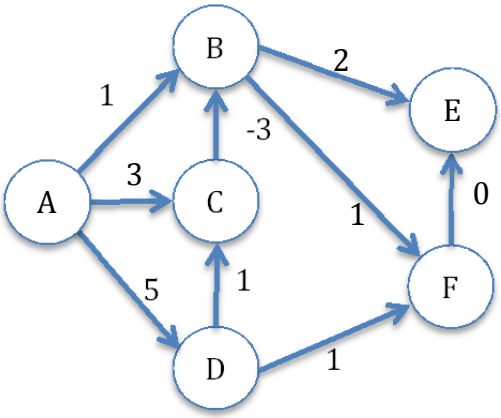
E. Suppose we want to use **Dijkstra's algorithm** to find the shortest paths from the source vertex "A" to the rest of the vertices in the graph given below. Fill out the following table to show what happens during the iterations of Dijkstra's algorithm. **[4 marks]**



vertex	Distance from source						
A							
B							
C							
D							
E							
F							
Best							

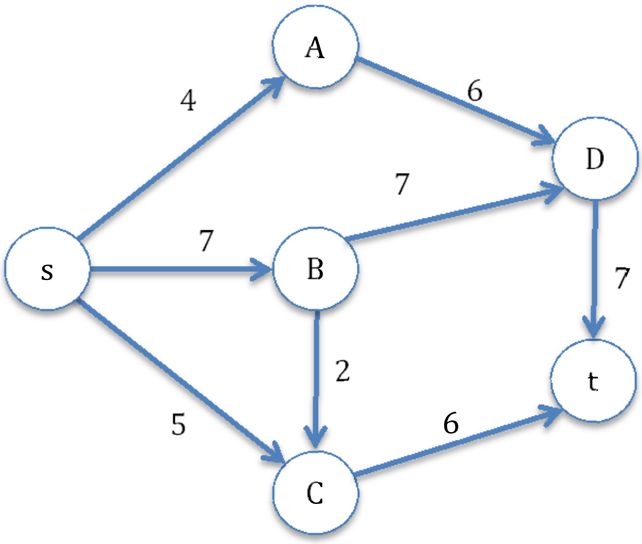
F. Draw the *shortest path graph* which includes just the edges selected by Dijkstra's algorithm in the previous question. **[1 mark]**

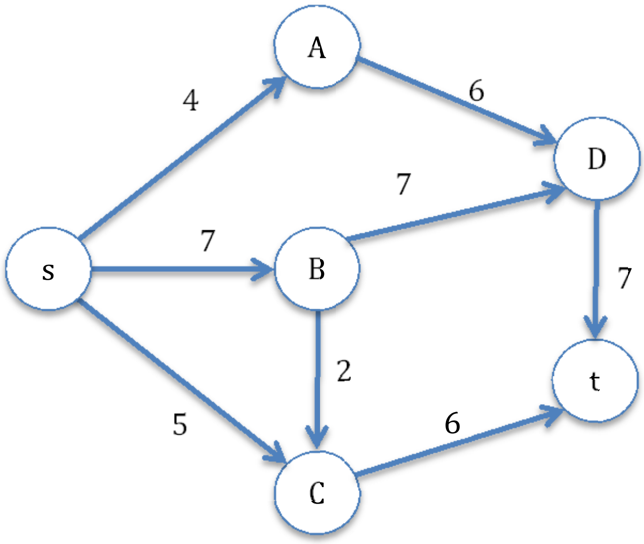
G. Suppose we want to use the **Bellman-Ford algorithm** to find the shortest paths from the source vertex "A" to the rest of the vertices in the graph given below. Fill out the following table to show what happens during the iterations of the Bellman-Ford algorithm. **[4 marks]**

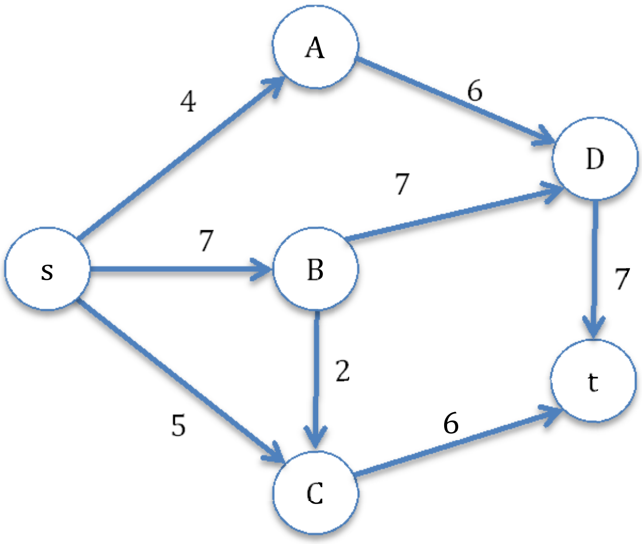


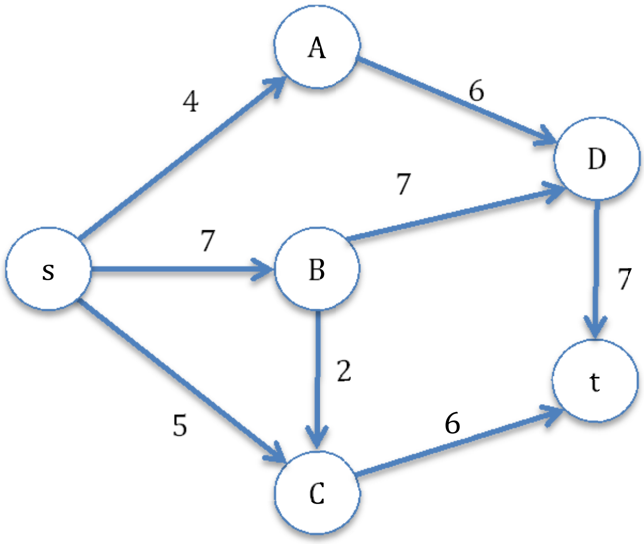
vertex	Distance from A					
A						
B						
C						
D						
E						
F						

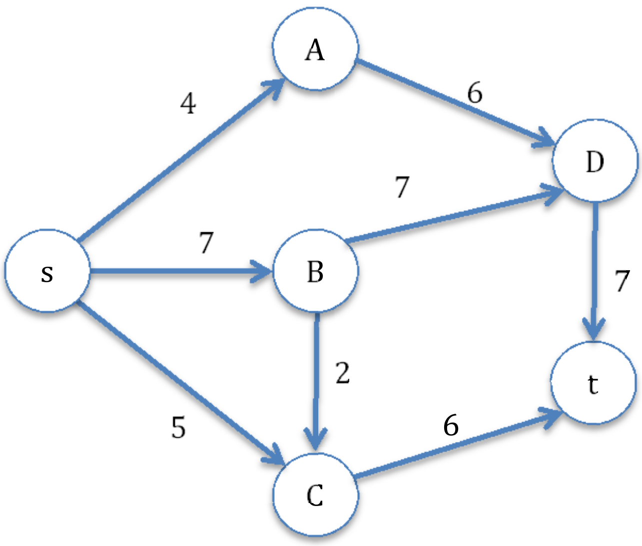
H. [Flow] Use the Ford-Fulkerson Algorithm to find a maximum flow from s to t in the following graph. Draw the augmenting graphs on top of the successive copies of the original graph. **[4 marks]**

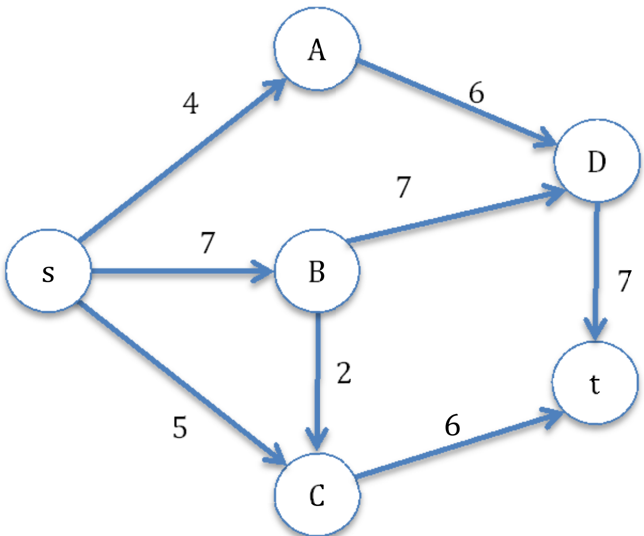












Question 5 [Algorithm Design] [14 marks]

For the following questions you need to write the pseudo-code of your algorithms.

A. [Greedy Algorithms] Give a greedy algorithm to find the longest path in a graph. Assume that the graph is represented by an adjacency list. **[7 marks]**

B. [Dynamic Programming] Write a dynamic programing algorithm to solve the following problem. Given a sequence of *n* real numbers A_1 ... A_n, determine the maximum sum of values for any contiguous subsequence. As an example, for the sequence 7, -1, 3, 5, 2, -6, 4 the answer is 16 which is the sum of the contiguous subsequence 7, -1, 3, 5, 2. **[7 marks]**