Office Use Only

MONASH University

Sample Exam

Faculty of Information Technology

EXAM CODES: **FIT1045**
TITLE OF PAPER: **INTRODUCTION TO ALGORITHMS AND PROGRAMMING – PAPER 1**
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 ☐ Monash Extension ☐ Sth Africa
☐ Parkville ☐ Other (specify)

During an exam, you must not have in your possession any item/material that has not been authorised for your exam. This includes books, notes, paper, electronic device/s, mobile phone, smart watch/device, calculator, pencil case, or writing on any part of your body. Any authorised items are listed below. Items/materials on your desk, chair, in your clothing or otherwise on your person will be deemed to be in your possession.

No examination materials are to be removed from the room. This includes retaining, copying, memorising or noting down content of exam material for personal use or to share with any other person by any means following your exam.
Failure to comply with the above instructions, or attempting to cheat or cheating in an exam is a discipline offence under Part 7 of the Monash University (Council) Regulations.

AUTHORISED MATERIALS

OPEN BOOK	☐ **YES**	☑ **NO**
CALCULATORS	☐ **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: __ __ __ __ __

Page		Mark
3		2
5		4
7		6
9		4
11		5
13		6
15		10

Page		Mark
17		5
19		5
21		8
23		8
25		10
27		7
Total:		80

Blank Page for Working

For Questions 1-6 circle only one letter for each question corresponding to the correct response.

Question 1 [1 mark]

What is printed by the following code?

```
def anon(n):
    if n % 2 == 1
        return 0
    else
        return 1 + anon(n//2)

print(anon(36))
```

(a) 5
(b) 4
(c) 2
(d) 1

Question 2 [1 mark]

Suppose you have the following function:

```
def secret(aList):
    val = 0
    for i in range(0, len(aList)//2):
        val += aList[i]*aList[len(aList)-i-1]
    return val
```

What is printed by the following code?

```
myList = [4, 3, 2, 1, 5]
print(secret(myList))
```

(a) 7
(b) 23
(c) 25
(d) 27

2

Blank Page for Working

Question 3 [1 mark]

The base case for Merge sort is a list of size?

(a) 0
(b) 1
(c) Both 0 and 1
(d) None of the above.

Question 4 [1 mark]

A function g(n) is said to be O(f(n)) if there exists constants k and L such that.

(a) g(n) > k*f(n) for all n < L
(b) g(n) < k*f(n) for all n > L
(c) g(n) < k*f(n) for all n < L
(d) g(n) > k*f(n) for all n > L

Question 5 [1 mark]

How many solutions are there for the 3 Queens problem?

(a) 0
(b) 1
(c) 2
(d) None of these.

Question 6 [1 mark]

An O(nlog(n)) algorithm always runs faster than an $O(n^2)$ algorithm. True or False? Why?

(a) False. For small n, constant factors may dominate the running time.
(b) True. O(nlog(n)) complexity is better than $O(n^2)$.
(c) True, but only if the input given to both algorithms is the same.
(d) False. $O(n^2)$ complexity is better than O(nlog(n)).

4

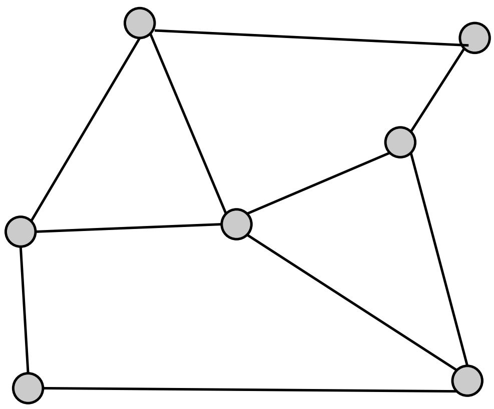
Blank Page for Working

Question 7 [2 + 2 + 2 = 6 marks]

(a) Give a definition of a process.

(b) Give a definition of a Hamiltonian cycle in a connected graph G.

(c) How many cliques of each size less than 5 are there in the following graph?



6

Blank Page for Working

Question 8 [1 + 1 + 1 + 1 = 4 marks]

This question is about *time complexity*. For each of the given Python functions, state the time complexity in big O notation and provide a brief explanation.

(a)

```
def total_func(n):
    total = 0
    for k in range(n):
        for j in range(n-k, 0, -1):
            total + k*j
    return total
```

(b)

```
def fraction_func(n):
    fraction = 1
    for k in range(100):
        for j in range(k):
            fraction = k + j + 1/fraction
    return fraction
```

(c)

```
def another_total_func(n):
    total = 0
    for k in range(n):
        total += k
    for k in range(10*n):
        total += num
    return total
```

(d)

```
def mid_func(n):
    low = 0
    high = n
    while low <= high:
        mid = (low + high)//2
        low = mid + 1
    return mid
```

4

Blank Page for Working

Question 9 [5 marks]

Write a Python function, **unique**, which takes as input a sorted list of strings, and returns **True** if the all the items in the list are unique. Otherwise the function should return **False**.

5

Blank Page for Working

Question 10 [1 + 3 + 2 = 6 marks]

The number comparisons for a version of Quick sort, **C(n)**, is defined by the following relations.

C(0) = 1 and C(n) = C(n//2) + n + 1,

(a) Give the value of **C(3)**.

(b) Write a recursive Python function which has as input a non-negative integer, **n**, and returns **C(n)**.

(c) State and explain the time complexity for your function.

6

Blank Page for Working

Question 11 [5 + 5 = 10 marks]

Consider the 4 Queen problem. Suppose we use the representation of a list of numbers where the kth number represents the row which the kth Queen, $\mathbf{Q_k}$, is in, e.g., [2, 1, 3, 0] would represent the following board.

Row 0				Q_3
Row 1		Q_1		
Row 2	Q_0			
Row 3			Q_2	

(a) Write a Python function, **lastQueenOk**, which has as input a list of numbers representing the positions of the Queens on 4x4 board. This function should return **True** if no Queen is attacking the Queen represented by the last number in the list, otherwise it should return **False**.

(b) Using the function, **lastQueenOk**, write a Python function**, isSolution**, which has as input a list of numbers representing the positions of the Queens on 4x4 board. This function should return **True** if no Queen is attacking any other Queen, otherwise it should return **False**.

10

Blank Page for Working

Question 12 [5 marks]

Suppose you have a collection of **n** items. All the items have the same weight, **w**, and you can choose at most **one** of each item. Write a Python function which is given as input, the capacity of the knapsack, **capacity**, a list (sorted in ascending order) of values, **values**, of each item, and the weight, **w**, and returns the maximum value that the knapsack can hold.

5

Blank Page for Working

Question 13 [5 marks]

Write a Python function, **duplicate,** which takes two lists sorted in ascending order) as input and returns a list of items that appear in both lists.

5

Blank Page for Working

Question 14 [6 marks]

Insert the following numbers, in the order they appear, into a Heap. You are allowed to choose whether the Heap is a min-Heap or a max-Heap.

10 6 11 -6 13 2

Show the Heap after each number has been inserted. The answer should consist of **6 Heaps.**

Question 15 [2 marks]

Insert the following numbers, in the order they appear, into Binary Search Tree.

10 6 11 -6 13 2

Show the Binary Search Tree after all the numbers have been inserted.

8

Blank Page for Working

Question 16 [2 + 6 = 8 marks]

Consider the problem of finding all the permutations of N different items.

a) Describe how a partial solution can be represented as list of numbers.

b) Show how you could use backtracking to solve this problem, when N = 3.

In particular, show a search tree and indicate on your diagram for each position the corresponding partial solution (represented as a list of numbers).

8

Blank Page for Working

Question 17 [4 + 3 + 3 = 10 marks]

(a) Explain why if a NP-complete problem could be solved in polynomial time, then P = NP.

(b) Give **3** examples of NP problems given in lectures and state their certificates.

(c) Describe the Halting problem.

10

Blank Page for Working

Question 18 [1 + 1 = 2 Marks]

(a) Give an example of a sorting method that has linear time complexity is the best case.

(b) Give an example of a sorting method that uses divide and conquer and has quadratic time complexity in the worst case.

Question 19 [5 Marks]

Write a Python function, **isVertexCover,** that checks whether a list of vertices, **vertexList,** is a vertex cover in a given graph.

The graph is represented as an adjacency matrix, **graphTable,** where **graphTable[j][k] = 1** if vertex **j** is adjacent to vertex **k**. The function, **isVertexCover**, takes **vertexList** and **graphTable** as input and returns **True** if the vertices in **vertexList** form a vertex cover of the graph represented by **graphTable**, and returns **False** otherwise.

7

END OF EXAM