Office Use Only

MONASH University

Semester Two 2016
Examination Period

Faculty of Information Technology

EXAM CODES: **FIT1008**

TITLE OF PAPER: **INTRODUCTION TO COMPUTER SCIENCE - PAPER 1**

EXAM DURATION: 3 hours writing time

READING TIME: 10 minutes

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

☐ Berwick	**X** Clayton	**X** 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**	**X NO**
CALCULATORS	☐ **YES**	**X NO**
SPECIFICALLY PERMITTED ITEMS	☐ **YES**	**X NO**
if yes, items permitted are:		

Page	Marks	Page	Marks
3		23	
5		25	
7		27	
9		29	
11		31	
13		33	
15		35	
17		37	
19		39	
21		**Total**	

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

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

This page intentionally left blank, use if needed but it will not be marked.

Question 1 [10 marks]

In this part you are required to answer the following short questions. Your answer should be concise. As a guideline, it should require no more space than the space that is provided.

(1) In MIPS, how many bits are required to store a word?

---

(2) In MIPS, how many bytes are required to store an array of 6 integers?

---

(3) Recursion is usually memory intensive because... (**Hint:** Use your MIPS knowledge)

---

(4) In the worst-case time complexity scenario, Merge-sort outperforms Quick-sort. However, quick-sort is often a better choice because...

---

(5) The two main operations of a Stack ADT are?

5

This page intentionally left blank, use if needed but it will not be marked.

(6) How does a Circular Queue differ from a standard array-based Queue?

(7) List 3 container ADT's covered in the lectures

(8) A class variable is...

(9) The instance variables of a simple Node to support a Linked Structure are....

(10) By convention, a parameter `self` in a method definition refers to...

5

This page intentionally left blank, use if needed but it will not be marked.

Question 2 [5 marks = 3 + 2]

This question is about sorting. Consider the following implementation of `selection_sort`.

```
def selection_sort(a_list):
    n = len(a_list)
    for k in range(n-1, -1, -1):
        max_position = find_max(a_list, k)
        a_list[k], a_list[max_position] = a_list[max_position], a_list[k]
```

(a) Using Python, define the function `find_max(a_list, limit_index)` that completes the implementation.

(b) Is Selection sort as implemented above stable? Explain your answer.

5

This page intentionally left blank, use if needed but it will not be marked.

Question 3 [8 marks = 2 + 2 + 2 + 2]

This question is about time complexity. For algorithms (a) to (d) express their Big-O notation time-complexity in the best and worst case. Provide a short explanation in each case. No explanation means no marks.

(a)

```
def algorithm_a(a_list):
    n = len(a_list)
    for k in range(n-1):
        a = k
        for i in range(k+1, n):
            if a_list[i] < a_list[a]:
                a = i
        a_list[k], a_list[a] = a_list[a], a_list[k]
```

Best time complexity: _____. Worst time complexity: _____
Explanation:

(b)

```
def algorithm_b(a_list):
    n = len(a_list)
    for k in range(0, n-1):
        position = 0
        for i in range(k, -1, -1):
            if a_list[i] == a_list[position]:
                break
            else:
                position += 1
        a_list[k], a_list[position] = a_list[position], a_list[k]
```

Best time complexity: _____. Worst time complexity: _____

Explanation:

4

This page intentionally left blank, use if needed but it will not be marked.

(c)
```
def algorithm_c(a_list):
    return a_list[-1]
```

Best time complexity: _____. Worst time complexity: _____

Explanation:

(d)
```
def algorithm_d(a_list, item):
    a = 0
    b = len(a_list) -1
    while a <= b:
        c = (a+b)//2
        if a_list[c] == item:
            return c
        elif a_list[c] > item:
            b = c - 1
        else:
            a = c + 1
    return -1
```

Best time complexity: _____. Worst time complexity: _____

Explanation:

This page intentionally left blank, use if needed but it will not be marked.

Question 4 [7 marks = 2 + 2 + 3]

This question is about Stacks. Consider the partial implementation of a Stack ADT below:

```
class Stack:
    def __init__(self, size):
        assert size > 0, "size should be positive"
        self.array =  size * [None]
        self.count =  0
        self.top = -1

    def is_full(self):
        return self.count > = len(self.array)

    def is_empty(self):
        return self.count == 0
```

(a) Implement the method `push(self, item)` using an assertion to check for the precondition.

(b) Implement the method `pop(self)` using an assertion to check for the precondition.

4

This page intentionally left blank, use if needed but it will not be marked.

(c) Consider the method `factorial` below, which relies on recursion.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

Provide a version of `factorial` which uses the Stack implementation to replace recursion.

3

This page intentionally left blank, use if needed but it will not be marked.

Question 5 [6 marks = 2 + 2 + 2]

This question is about linked structures. Consider the following partial implementation of a linked SortedList.

```
class Node:
    def __init__(self, item=None, link=None):
        self.item = item
        self.next = link

class SortedList:
    def __init__(self):
        self.head = None
        self.count = 0

    def _getnode(self, index):
        assert 0 <= index <= self.count, "index out of bounds"
        node = self.head
        for _ in range(index):
            node = node.next
        return node
```

(a) Define the method `add(self, item)`, which adds one item to the list keeping the list sorted.

(b) What is the best and worst-case time complexity of a correct and efficient implementation of `add(self, item)` for this data type. Explain your answer.

This page intentionally left blank, use if needed but it will not be marked.

(c) Define the method `__next__(self)`, of the Iterator below, which is intended to go through all elements of the Sorted List defined above.

```
class SortedListIterator:
    def __init__(self, head):
        self.current = head
    def __iter__(self):
        return self
```


2

This page intentionally left blank, use if needed but it will not be marked.

Question 6 [6 marks = 3 + 3]

This question is about recursion.

(a) The greatest common divisor (GCD) of two integer numbers is the largest positive integer that divides the numbers without a remainder. Convert the following iterative version of the GCD algorithm into a recursive algorithm.

```
def gcd(a, b):
    while(b != 0):
        r = a%b
        a = b
        b = r
    return a
```

(b) Provide a tail-recursive version of the following algorithm::

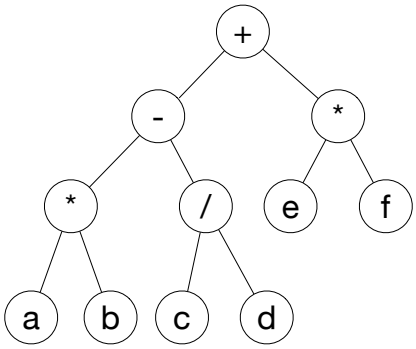
```
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n-2) + fib(n-1)
```


6

This page intentionally left blank, use if needed but it will not be marked.

Question 7 [10 = 1 + 2 + 2 + 5 marks]

This question is about Binary Trees and Binary Search Trees. Consider the graph below, which represents an expression tree:



(a) What is the infix arithmetic expression given by this Binary Tree?

(b) List the sequence of characters as they occur when you traverse the tree above in pre-order?

(c) List the sequence of characters as they occur when you traverse the tree above in post-order?

5

This page intentionally left blank, use if needed but it will not be marked.

(d) For the BinarySearchTree data type defined below, write down the recursive method `_insert_aux(self, current, key, item)`:

```
class BinarySearchTreeNode:
    def __init__(self, key, item=None, left=None, right=None):
        self.key = key
        self.item = item
        self.left = left
        self.right = right


class BinarySearchTree:
    def __init__(self):
        self.root = None

    def insert(self, key, item):
        self._insert_aux(self.root, key, item)
```


5

This page intentionally left blank, use if needed but it will not be marked.

Question 8 [10 marks = 3 + 3 + 4]

This question is about Heaps. Consider the partial implementation of a Max Heap.

```
class Heap:
    def __init__(self):
        self.count = 0
        self.array = [None]

    def __len__(self):
        return self.count

    def add(self, item):
        if self.count + 1 < len(self.array):
            self.array[self.count + 1] = item
        else:
            self.array.append(item)
        self.count += 1
        self.rise(self.count)

    def swap(self, i, j):
        self.array[i], self.array[j] = self.array[j], self.array[i]

    def get_max(self):
        item = self.array[1]
        self.swap(1, self.count)
        self.count -= 1
        self.sink(1)
        return item

    def sink(self, k):
        while 2*k <= self.count:
            child = self.largest_child(k)
            if self.array[k] >= self.array[child]:
                break
            self.swap(child, k)
            k = child
```

(a) Implement the method `rise(self, k)` which complements the `add` function.

3

This page intentionally left blank, use if needed but it will not be marked.

(b) Implement the method `largest_child(self, k)` which complements the `sink` function

(c) Using only the methods defined above define a function `get_minimum(a_max_heap)` that returns the minimum element of the Heap in $O(N)$. The parameter of the function is assumed to be a Max-Heap. Your method **can** modify the Heap if necessary, but only through the operations of the Data Type (i.e., you should not access instance variables directly).

This page intentionally left blank, use if needed but it will not be marked.

Question 9 [9 marks = 3 + 3 + 3]

This question is about Hash Tables. Keep answers short using the space provided only.

(a) Explain how quadratic probing is used to resolve collisions in a HashTable.

(b) Explain how separate chaining is used to resolve collisions in a HashTable.

(c) If you were given a perfect hash function, would collision handling be necessary? Explain why/why not.

9

This page intentionally left blank, use if needed but it will not be marked.

Question 10 [11 marks]

Translate to MIPS faithfully using only the instructions available in the reference sheet and following the function calling convention discussed in the lectures.

Python Code	MIPS Code

`a = 5`
 	
`a_abs = 0`	

`def my_function(x):`
 	
`  if x > 0:`
`  return x`
 	
`  else:`
 	
`  return -x`	

`a_abs = my_function(a)`
 	
`print(a_abs)`	

11

This page intentionally left blank, use if needed but it will not be marked.

Question 11 [12 marks = 4 + 4 + 4]

For each situation described in the rows of the table below, choose one of the following structures: Linked List (1), Array-based List (2), SortedList Array-based (3), or Sorted Linked-List (4). Explain briefly the reason behind your choice in the space provided.

Situation	**Choice of Data Type**	**Explanation**
A post office needs to store in a list a record for each packet being processed. Postal demand is volatile so the number of packages arriving each day is unpredictable. It is not necessary to keep track of the order.		
A University needs to keep a list of students. Demand is predictable so the approximate size of the list is known and after enrolments not a lot of changes are needed in terms of additions or deletions. The students do not need to be sorted.		
A University needs to keep a list of employees. The approximate size of the list is known and new items or removals happen infrequently. The critical functionality to be provided is search by name.		

This page intentionally left blank, use if needed but it will not be marked.

Question 12 [6 marks = 1 × 6]

In MIPS, the function calling convention has the following steps. For each step, explain in the space provided **why** this action is required by the convention.

(1) Caller saves temporary registers on the stack.

(2) Caller passes arguments on to the stack.

(3) Caller calls function using `jal`.

3

This page intentionally left blank, use if needed but it will not be marked.

(4) Callee saves `ra` and `fp` on the stack.

(5) Callee copies `sp` to `fp`.

(6) Callee allocates local variables on the stack.

END OF EXAM.

3