MONASH University

Office Use Only

Semester Two 2014
Examination Period

Faculty of Information Technology

EXAM CODES: **FIT1008**
TITLE OF PAPER: **COMPUTER SCIENCE**
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
☐ Parkville ☐ 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, or attempting to cheat or cheating in an exam is a discipline offence under Part 7 of the Monash University (Council) Regulations.

No exam paper or other exam materials are to be removed from the room.

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		5
5		10
7		6
9		4
11		6
13		4
15		4

Page		Mark
17		10
19		10
21		7
23		6
25		8

Total:		**80**

Blank Page

Question 1 (4 + 1 = 5 marks)

(a) Write a function, **`insertion_sort`**, in Python which takes as input a list, **`the_list,`** of numbers and sorts this list into **increasing** order using insertion sort.

(b) Explain why the following algorithm does not implement a stable sorting method.

```
def unstable_sort(the_list):
    n = len(the_list)
    for _ in range(n-1):
        for j in range(n-1):
            if the_list[j] >= the_list[j+1]:
                the_list[j], the_list[j+1] = the_list[j+1], the_list[j]
```

5

Question 2 (6 + 2 + 2 = 10 marks)

(a) This question is intended to test your skills at programming with *queues* and *iterators.*

Consider a **`Queue`** class which implements a queue using some data structure (you do not need to know which one) and defines the following methods:

```
__init__()
append(item)
serve()
is_empty()
__iter__()
```

Define a new method (outside this class, so you have no idea how the queue is implemented and, therefore, can ONLY use the above methods):

`pushBack(queue)`

which takes as input a queue and returns a new queue where every negative integer has been pushed to the back of the queue preserving the same order.

For example, if the input queue holds the values **5, 8, -9, 10, -3, -7, 4,** where **5** is at the front of the queue, then the method should return a new queue with the characters in the following order:

5, 8, 10, 4, -9, -3, -7

Your method is *NOT* allowed to modify the input queue. Of course, if the input queue is empty, the output queue should also be empty.

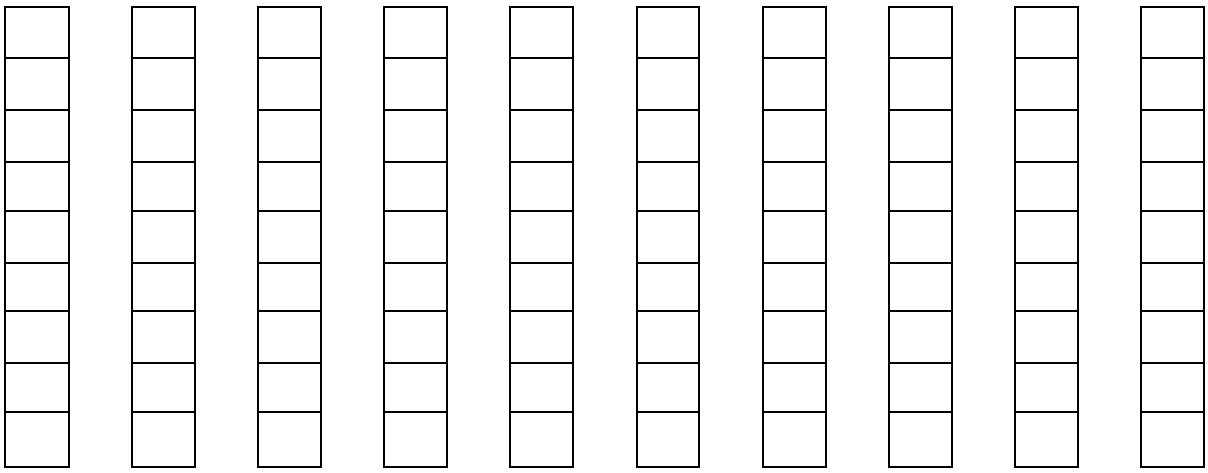
(b) This question is about *understanding code*. Consider a stack data type provided by the **`Stack`** class which is implemented using some data structure (you do not need to know which one) and defines the following methods:

```
__init__()
pop()
push(item)
is_empty()
```

Consider the following function that uses the above methods:

```
def mystery(list):
    the_stack = Stack()
    for item in list:
        if item > 0:
            the_stack.push(item)
        elif item == 0:
            for i in range(2):
                the_stack.push(i)
        elif not the_stack.is_empty():
            the_stack.pop()
        # HERE
```

Use the sequence of stacks below to draw the elements on the stack every time the line **# HERE** is reached during the execution of **`mystery([4, 9, -3, 0, 0, -3]).`**



(c) This question is about *choosing data structures*. Several data types (such as stacks, queues, and lists) can be implemented using arrays or using linked nodes. Explain an advantage and a disadvantage of the array implementation compared to a linked node implementation.

10

Blank Page

Question 3 (6 marks)

This question about *array implementation of lists.* Consider the class **`SortedList`** which implements a **sorted list data type** using an array

```
class SortedList:
     def __init__(self, size):
         assert size > 0, "size should be positive"

         self.the_array = size*[None]
         self.count = 0

     def __len__(self):
         return self.count

     def is_empty(self):
         return len(self) == 0

     def is_full(self):
         return len(self) >= len(the_array)
```

Define a method for this class **`insert(self, the_item)`** which adds a new item, **`the_item`**, to the list in the correct position. If the list is full the method should return **`False`**, else it should return **`True.`**

6

Blank Page

Question 4 (1 + 1 + 1 + 1 = 4 marks)

This question is about *time complexity*. For each of the given Python functions, identify the time complexity for both best and worst case by providing an explanation. (*No explanation means no marks.*)

(a)
```
def product_func(n):
    product = 1
    for _ in range(n):
        for num in range(5):
            product *= num
```

(b)
```
def total_func(n):
    total = 0
    for _ in range(n):
        for num in range(n):
            total += num
```

(c)
```
def another_total_func(n):
    total = 0
    for num in range(n):
        total += num
    for num in range(n+1):
        total += num
```

(d)
```
def division_func(n):
    product = 1
    while n > 1:
        product *= 2
        n //= 2
```

4

Blank Page

Question 5 (6 marks)

This question is about *linked structures*. Consider the two classes **`Node`** and **`List`** as seen in the lectures, which define a **list data type** implemented using a linked structure:

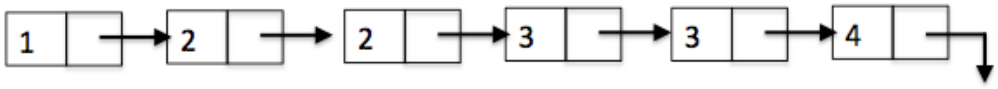
```
class Node:
    def __init__ (self, item = None, link = None):
        self.item = item
        self.next = link

class List:
    def __init__ (self):
        self.head = None
    def is_empty (self):
        return self.head is None
```

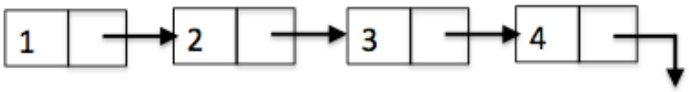
Add the following method to the **`List`** class:

```
def remove_duplicates(self):
```

which removes all the duplicates of each item in a sorted list. For example, consider a **`List`** object **`a_list`** whose **`a_list.head`** points to the first node of the list whose **`item`** is **1**.



After calling **`a_list.remove_duplicates(), a_list`** should contain only the unique items.



6

Blank Page

Question 6 (4 marks)

This question is about *iterators*. Define a **`PrimeIterator`** iterator class, where given an integer **n**, an instance of this class would produce the first prime starting from **n**, then the second prime, the third prime, and so on indefinitely.
For example:

```
>>> it = PrimeIterator(3)
>>> next(it)
3
>>> next(it)
5
>>> next(it)
7
>>> next(it)
11
```

In other words, define the following three methods for the class **`PrimeIterator`**: **`__init__`**, **`__iter__`** and **`__next__`**.

4

Blank Page

Question 7 (2+2 = 4 marks)

(a) Consider the following code for the binary search algorithm, where the **`the_array`** is an array and target is the item you are searching for in **`the_array`**. Find the bugs in code and describe how you would fix them.

```
def binary_search(the_array, target):
     low = 0
     high = len(the_array)

     while (low < high):
         mid = (low + high) // 2

         if (the_array[mid] == target):
             return mid
         elif (target < the_array[mid]):
             high = mid
         else:
             low = mid

     return -1
```

(b) Explain the best and worst time complexity for the binary search algorithm.

4

Blank Page

Question 8 [8 + 2 = 10 marks]

(a) This question is about *MIPS programming and understanding function calls*. Translate the following Python code faithfully into MIPS assembly language. Make sure you follow the MIPS function calling and memory usage conventions.

Python Code	MIPS Code
`def collatz(n):`	
`    if n % 2 == 0:`	
`        return n // 2`	
`    return 3*n + 1`	

.

(b) Explain why a recursive function may use more memory than an iterative function.

10

Question 9 (6 + 4 = 10 marks)

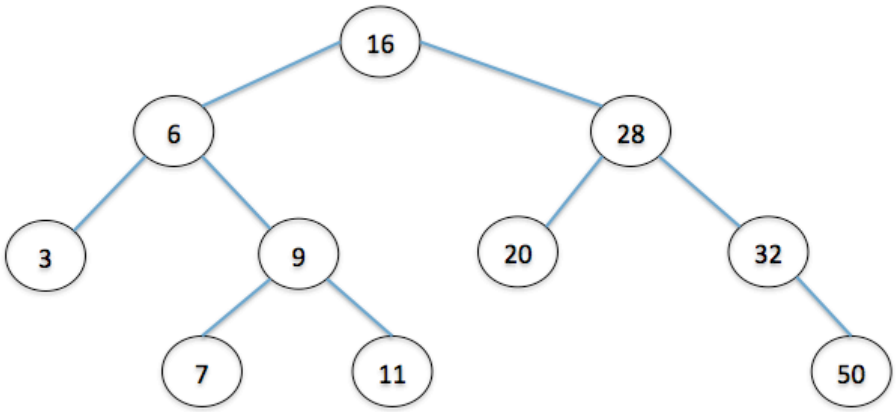
(a) This question is about *recursive programming* and *linked structures.* Consider the two classes **`BinarySearchTreeNode`** and **`BinarySearchTree`** which define a binary search tree data type implemented using linked nodes, and which are defined as follows:

```
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
```

Define a recursive method **`collect_leaves(self)`** inside the **`BinarySearchTree`** class that collects the keys of all the leaf nodes (i.e. nodes without children) in the binary search tree and returns them inside a Python list in increasing order.

For example, if the binary search tree is:



the method will return the Python list **`[3,7,11,20,50].`** If the binary search tree is empty the list returned will also be empty.

IMPORTANT: you are defining the method inside the class **`BinarySearchTree`** and therefore you can access anything inside that class. Also, you are using Python lists and therefore you can use any operation that Python makes available on them, such as **`+`**.

Write your answer of this part on the next page

Write your answer for Question 9(a) here

(b) This question is about *recursive sorts*. Both merge sort and quick sort are divide and conquer algorithms, i.e., they divide the original problem into sub-problems (hopefully of roughly equal size), they solve each sub-problem independently, and then combine the solutions to solve the original problem.

What are the main differences between merge sort and quick sort in terms of the amount of effort put into the dividing and combining operations mentioned above?

10

Blank Page

Question 10 (4 + 2 + 1 = 7 marks)

This question is about *Binary Trees.*

(a) Explain how a Binary Search Tree being balanced or unbalanced affects the best and worst time complexity for searching for an item.

(b) Construct an expression tree for the following arithmetic expression.

4 + 2 * 3 - 5 * 8

(c) Traverse the Expression Tree you constructed in Part (b) in preorder and write the values of the nodes as you visited them.

7

Blank Page

Question 11 (2 + 4 = 6 marks)

This question is about *heaps* and in this question the Heap is a ***max Heap***.

(a) What is the minimum number of elements that must be moved during a "retrieve the maximum element" operation on a heap? Give an example of a heap with 7 elements for which a "retrieve the maximum element" operation will require this minimum number of moves.

(b) Provide an implementation of the method **`delete(self, k)`** that deletes from a **Heap** the element at position ***k***. You can assume that the heap's elements are numbers stored in an array called **`array`** (with the root at position 1), that you have an instance variable **`count`**, and that the **Heap** has the following methods:

```
sink(self, k)
rise(self, k)
swap(self, i, j).
```

6

Blank Page

Question 12 (2 + 2 + 4 = 8 marks)

This question is about *hash tables*.

(a) Why is Quadratic Probing better than Linear Probing when inserting a new element? Explain in terms of the probe chain (*no explanation means no marks*).

(b) Describe an operation that is more efficient to implement with a Binary Search Tree than a Hash Table?

(c) Consider the class **`Hash`** which has the instance variables **`array, tablesize`**, and **`count`**, and the following methods:

```
__init__()
hash(key)
rehash()
```

Using Linear Probing, define a method **`__setitem__(self, the_key, data)`** which does the following:

- If there is an entry in the hash table with key, **`the_key`**, then it changes the value associated with **`the_key`** to **`data`**.
- If there is not entry with key, **`the_key`**, then an entry is inserted in the table with key, **`the_key`**, and value, **`data`**.

8

End of Exam