1602/159.172 2Hr
MAN/ALB/HEBT
Distance/Internal

MASSEY UNIVERSITY
MANAWATU AND ALBANY CAMPUSES AND HEBEI UNIVERSITY OF TECHNOLOY

EXAMINATION FOR
159.172 COMPUTATIONAL THINKING AND ALGORITHMS

Semester Two 2016

Time allowed**:** **TWO (2)** hours

Attempt all parts of **ALL THREE (3)** questions

Question 1 is worth 20 marks
Question 2 is worth 10 marks
Question 3 is worth 15 marks

This examination contributes 40% to the final assessment

All answers to be written in the blue answer booklet provided.

Calculators not permitted

This page is intentionally left blank.

QUESTION 1: (20 marks)

Choose one answer for each of the following 20 multi-choice questions in the blue answer booklet provided.

.

1 mark for each correct answer.

1. What will the following code display?

```
>>> vowels = ['a, 'e', 'i', 'o', 'u']
>>> vowels[1:3]
```

(a) `['a', 'e', 'i']`
(b) `['e', 'i']`
(c) `['e, 'i', 'o']`
(d) `['a', 'e']`

2. What is the list generated by the list comprehension given here?

```
[x+2*y for x in range(3) for y in range(4)]
```

(a) `[0, 2, 4, 6, 1, 3, 5, 7, 2, 4, 6, 8]`
(b) `[11]`
(c) `[3, 5, 7, 4, 6, 8, 5, 7, 9, 6, 8, 10]`
(d) `[0, 2, 4, 1, 3, 5, 2, 4, 6, 3, 5, 7]`

3. Suppose that the predicate even is defined as:

```
def even(x):
    return x % 2 == 0
```

Which of the following will produce the list `[0, 2, 4, 6, 8]`?

(a) `map(even, range(9))`
(b) `filter(even, range(9))`
(c) `[even(x) for x in range(9)]`
(d) `[x for x in even(range(9))]`

4. If a list is already sorted in ascending order, a modified sequential search algorithm can be used that compares against each element in turn, stopping either if the target value is found or if the current list element exceeds the target value. What is the worst-case number of comparisons of this “short” sequential search on a sorted *n*-element list?

(a) log *n*
(b) 2*n*
(c) *n*
(d) *n* - 1

5. Which of these function definitions will correctly produce the output shown here?

```
>>> countdown(3)
3
2
1
0
```

(a)
```
def countdown(n):
  print(n)
  if n == 0:
    return
  countdown(n - 1)
```

(b)
```
def countdown(n):
  if n == 0:
    return
  print(n)
  countdown(n - 1)
```

(c)
```
def countdown(n):
  if n == 0:
    return
  countdown(n - 1)
  print(n)
```

(d)
```
def countdown(n):
  countdown(n - 1)
  print(n)
  if n == 0:
    return
```

6. Let n have the value 5. What is the sequence of values printed out by the following code fragment?

```
k = 1
while (k < n):
   j = n//2
   while (j >= 1)
      print j
      j = j//2
   k = k + 1
```

(a) **2, 1, 2, 1, 2, 1, 2, 1**
(b) **2, 1, 0, 2, 1, 0, 2, 1, 0, 2, 1, 0**
(c) **2, 2, 2, 2**
(d) **2**

7. In terms of *n*, what is the running time of the code fragment given in question 6 above?

(a) $\Theta(n^2)$
(b) $\Theta(n \log n)$
(c) $\Theta(n)$
(d) $\Theta(1\log n)$

8. For the class definitions given here, what does **`mysteryMethod()`** achieve?

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

class LinkedList:
  def __init__(self):
    self.head   = None

  def mysteryMethod(self, cargo):
    node = Node(cargo)
    if self.head == None:
      self.head = node
    else:
      tracker = self.head
      while tracker.next is not None:
        tracker = tracker.next
        tracker.next = node
```

(a) adds a new node containing **`cargo`** to the beginning of the list
(b) adds a new node containing **`cargo`** to the end of the list
(c) adds a new node containing **`cargo`** in the second to last position in the list
(d) sets **`tracker`** to reference the last node in the list

9. Consider the function definition given here.

```
def printTree(tree):
  if tree == None: return
  print tree.cargo,
  printTree(tree.left)
  printTree(tree.right)
```

This function performs …

(a) a pre-order traversal of a tree.
(b) a post-order traversal of a tree.
(c) an in-order traversal of a tree.
(d) a reverse-order traversal of a tree.

10. Using the definition for the **`printTree()`** function from question 9, and the class definition given here:

```
class Tree:
  def __init__(self, cargo, left=None, right=None):
    self.cargo = cargo
    self.left  = left
    self.right = right

  def __str__(self):
    return str(self.cargo)
```

What will the following output?

```
>>> tree = Tree('+', Tree(1), Tree('*', Tree(2), Tree(3)))
>>> printTree(tree)
```

(a) **`1 2 3 * +`**
(b) **`1 + 2 * 3`**
(c) **`3 * 2 + 1`**
(d) **`+ 1 * 2 3`**

11. Consider the following function definition:

```
def histogram(s):
  d = dict()
  for c in s:
    d[c] = d.get(c, 0) + 1
  return d
```

What will the following display?

```
>>> hist = histogram('parrot')
>>> print hist
```

(a) **`{'a': 1, 'p': 1, 'r': 2, 't': 1, 'o': 1}`**
(b) **`{1: ['a', 'p', 't', 'o'], 2: ['r']}`**
(c) **`{1 :'a', 1 :'p', 2 :'r', 1 :'t', 1 :'o'}`**
(d) **`{1 :'p', 2 :'a', 3 :'r', 4 :'t', 5 :'o'}`**

12. Consider the function definition given here:

```
def draw_snowmen(screen, x, y):
  if x > 1000:
     return
  else:
     draw_snowman(screen, x, y)
     draw_snowmen(screen, x+100, y)
```

How many snowmen will be drawn by this call to the function?

```
draw_snowmen(screen, 10, 200)
```

(a) **1**
(b) **0**
(c) **9**
(d) **10**

13. Suppose we change the condition in the first line of the function definition:

```
def draw_snowmen(screen, x, y):
  if x < 1000:
     return
  else:
     draw_snowman(screen, x, y)
     draw_snowmen(screen, x+100, y)
```

Which of the following is true of this new version of the function?

(a) Any call to the function will draw exactly one snowman.
(b) Any call to the function will not terminate.
(c) The function call **`draw_snowmen(screen, 10, 200)`** will not terminate.
(d) The function call **`draw_snowmen(screen, 10, 200)`** will not draw any snowmen

14. The following array represents a max-heap (largest value at the root). How many descendants does the node containing the value 24 have?

104	71	24	66	27	23	8	5	32	25	18	22

(a) 3
(b) 6
(c) 5
(d) 1

15. Assuming the data in a min-heap (smallest value at the root) are all distinct, what are the possible locations for the **largest** element?

    (a) any node in the bottom level only
    (b) any node without a child
    (c) the leftmost node in the bottom level
    (d) the rightmost node in the bottom level

16. Which value is returned by **`Mystery`**, if the function is called with a value of 1?

```
def Mystery(n):
        if (n < 5):
            return Mystery(n + 1) + 2 * Mystery(n + 2) + 1
        else:
            return 0
```

    (a) 5
    (b) 10
    (c) 15
    (d) 20

17. When does a collision occur during hashing?

    (a) When all keys map to different hash locations.
    (b) When two or more keys map to the same hash location.
    (c) When a key is mapped to an empty hash location.
    (d) When the hash table does not contain any keys.

18. Which of the following strategies should **not** be used to resolve collisions in hash tables?

    1. Let each cell of the hash table be a linked list that contains all key-value pairs hashed into this cell.
    2. Override the value of a cell if a new value is hashed into the cell.
    3. Take the next available cell (in successive order) if the hash function yields a cell that is already occupied.
    4. Dismiss the new value if its key leads to a cell that is occupied.

    (a) 2 and 3
    (b) 1 and 4
    (c) 2 and 4
    (d) 1 and 3

19. Which of the following statements is **not true** for topological sort?

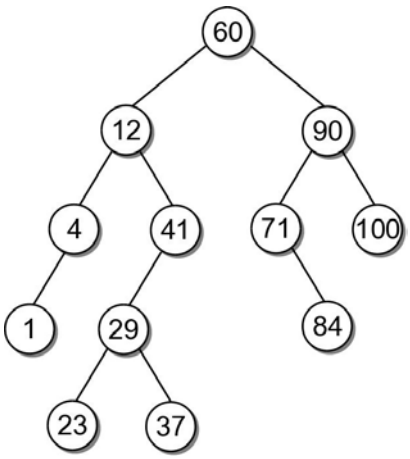
    (a) Topological sort uses a prerequisite structure to sort items.
    (b) Each prerequisite structure results in exactly one topological sort.
    (c) Topological sort does not always involve pieces of clothing.
    (d) Depth-first search can be used to construct a topological sort.

20. Which of the following statements is **true** for the Knuth-Morris-Pratt text search algorithm?

    (a) The algorithm uses fingerprints to speed up the search.
    (b) The algorithm is based on a shift table.
    (c) The algorithm uses an occurrence heuristic to find a match.
    (d) All of the above

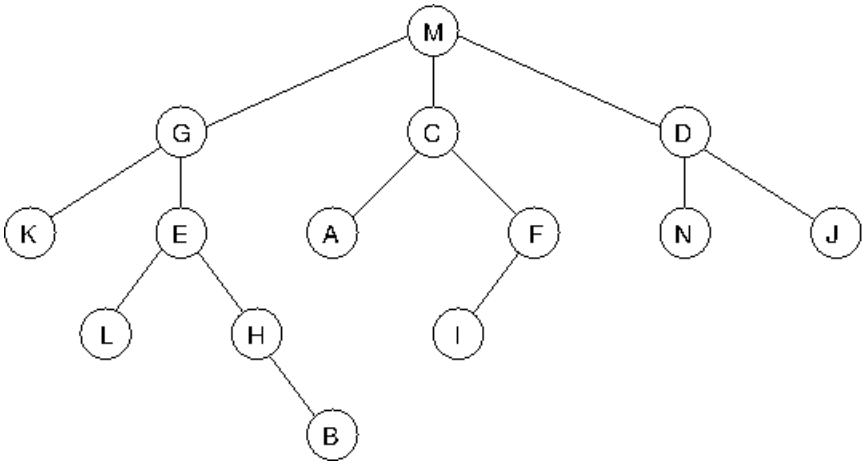
QUESTION 2: (10 marks)

1. Describe the four steps necessary to delete key 12 from the following search tree:



[5 marks]

2. List the nodes in the order they are visited when a **depth-first search** for **F** is performed on the following graph. Assume that successor nodes are considered from left to right.



[3 marks]

List the nodes in the order they are visited when a **breadth-first search** for **F** is performed on the graph above.

[2 marks]

QUESTION 3: (15 marks)

1. Consider the function definition given here, which is intended to calculate the value x^y, where **x** is the base and **y** is the exponent. Both the base and the exponent may be positive or negative integers.

   Recall that $x^0 = 1$ and that $x^{-m} = 1/x^m$

```
def power(base, exponent):
      if exponent < 0:
         return 1.0/power(base, -exponent)
      else:
         return base*power(base, exponent-1)
```

   Explain what is wrong with this function definition.

   [2 marks]

   Rewrite the function definition so that it works correctly.

   [3 marks]

2. Write a **count** method, for the **LinkedList** class given below, that takes a cargo item as an argument and returns the number of times the cargo item occurs in the list.

```
def count(self, cargo_item):

class Node:
  def __init__(self, cargo=None, next=None):
    self.cargo = cargo
    self.next  = next

class LinkedList:
  def __init__(self):
    self.length = 0
    self.head   = None

  def addFirst(self, cargo):
    node = Node(cargo)
    node.next = self.head
    self.head = node
    self.length = self.length + 1
```

   [5 marks]

3. Write a recursive function **no_duplicates** that takes a list of numbers and returns **True** if each number in the list is unique (no duplicates) and **False** otherwise.

   Write your function using Python syntax, minor syntax errors will not be penalised. Make sure that your function is recursive. What is the base case for the problem?

   [5 marks]

+ + + + + + + +