Semester I Examinations 2019/2020

Exam Code(s)	1MAO2, 1MAI1
Exam(s)	MSc in Computer Science (Artificial Intelligence), MSc in Computer Science (Artificial Intelligence) – Online
Module Code(s)	CT5148
Module(s)	**Programming and Tools for Artificial Intelligence**, **Programming and Tools for Artificial Intelligence – Online**
Paper No.	1
External Examiner	Prof. Pier Luca Lanzi
Internal Examiner(s)	Prof. Michael Madden Dr. James McDermott *
Programme Coordinator(s)	Dr. Michael Schukat Dr. James McDermott
Instructions	**Answer ALL questions.** When writing code, comments and error-checking are not required except where explicitly stated.
Duration	2 Hours
Number of pages	5 (including this page)
Discipline	Computer Science
Requirements	
Release in Exam Venue	Yes ☒ No ☐
Release to Library	Yes ☒ No ☐

Question 1: Basic Python

(a) Your friend has found some good code on the internet, but accidentally discarded the indentation when copy-pasting. Fix the problem(s) by inserting appropriate indentation. **[5]**

```
def int_sqrt(x):
"""Approximate integer square root"""
if x < 0:
raise ValueError("Bad input")
else:
guess = 0
while (guess + 1) ** 2 <= x:
guess += 1
return guess
```

(b) Re-implement the following code in pure Python without `collections`. **[5]**

```
from collections import defaultdict
d = defaultdict(int)
dna = "gattaca"
for s in dna:
    d[s] += 1
```

(c) Explain *duck typing* in Python using an example. **[5]**

(d) Rewrite the following using `itertools`. What is the benefit of this change? **[5]**

```
xs = [0, 1, 2, 3]
ys = [False, True]
for x in xs:
    for y in ys:
        print(x, y, f(x, y)
```

(e) Describe the concept of *machine epsilon*. How could we go about finding the value of machine epsilon in Python? **[5]**

Question 2: Advanced Python

(a) Define *memoisation* and describe the properties a function must have for memoisation to be useful. **[5]**

(b) Suppose we have a *string* `s` containing an arithmetic expression in Python syntax, including constants and variables `x` and `y`. Write code which will create a function `f(x, y)`, as in the example below. **[5]**

```
>>> s = "x**2 + y + 1"
>>> # your code here
>>> f(3, 2)
12
```

(c) The following regular expression matches common email addresses. Show how we can use *grouping* to extract the *username* portion from a given email address `adr`, i.e. the portion before the `@` symbol. Show both the changes in the regular expression and the necessary Python code. **[5]**

```
p = r"[\w\.-]+@\w+(\.\w+)+$"
```

(d) Rewrite the following function as a generator (not a generator comprehension). What is the main benefit of using a generator over a function? **[5]**

```
def f(filename):
    result = []
    for line in open(filename):
        x = int(line)
        result.append(x**2)
    return result
```

(e) Rewrite the following function using a dictionary for dispatch and without `if` statements. **[5]**

```
def f(a, b):
    if a and not b:
        g0()
    else:
        if a:
            g1()
        elif b:
            g2()
        else:
            g3()
```

Question 3: Data Science

(a) Given a Numpy array named `X` with contents as below (left), use fancy indexing of `X` to give the array below (right). **[5]**

```
[[100 101 102 103]
 [110 111 112 113]
 [120 121 122 123]
 [130 131 132 133]]
```

```
[[111 112 113]
 [121 122 123]]
```

(b) Describe the rules for broadcasting compatibility in Numpy. Using these rules, say whether this code will work, and if so what is the result, and if not why not. **[5]**

```
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([[10], [100]])
a + b
```

(c) Given a Pandas `DataFrame` as shown, named `d`, what is the result when we call `d.groupby("y").mean()`? **[5]**

```
   x  y   z
0  a  a  10
1  a  b  50
2  b  a  20
3  b  b  60
```

(d) Consider an image of 100×100 pixels with 3 colour channels. What shape does it have? What shape would it have if converted to a tidy format? **[5]**

(e) Describe Scikit-Learn's `GridSearchCV`: what inputs does it require, what does it do, what is the result? **[5]**

Question 4: Tools and Applications

(a) Suppose we have an *electrocardiogram* (ECG) time-series signal of length 20 seconds, stored in a Numpy array. Describe (no need to provide code or calculations) how we can use basic Numpy operations to estimate the heart rate in bpm. **[5]**

(b) In an adjacency matrix representation for graphs, give an interpretation of the following properties: the row-sum for a given row; asymmetry in the matrix; non-zero values on the diagonal. **[5]**

(c) Describe how *topological sorting* of a graph can be applied in project planning. **[5]**

(d) Given the finite state machine defined below, write out the sequence of states and outputs which will occur when it is executed with input `0010011110`. **[5]**

```
# (state, input) -> (state, action) mapping
SISAs = {
    ("W", 0): ("W", "waiting for a task"),
    ("W", 1): ("A", "acting on a task"),
    ("A", 0): ("W", "finished a task"),
    ("A", 1): ("F", "shutting down"),
    }
start_state = "W"
end_states = {"F"}
```

(e) Consider the grammar below, where `<expr>` is the start symbol. Show how the sentence `not (x[0] or not x[1])` can be derived from it. What is the difference between a terminal and a non-terminal? **[5]**

```
<expr>  ::= (<expr> <biop> <expr>) | <uop> <expr> | <var> | <const>
<biop>  ::= and | or
<uop>   ::= not
<var>   ::= x[0] | x[1] | x[2]
<const> ::= True | False
```