Semester I Examinations 2021/2022

Exam Code(s)	1MAI1, 1CSD1, 1SPE1, 2SPE1
Exam(s)	MSc in Computer Science (Artificial Intelligence), MSc in Computer Science (Artificial Intelligence) - Online
Module Code(s)	CT5132, CT5148
Module(s)	**Programming and Tools for Artificial Intelligence**
Discipline	School of Computer Science
Paper No.	1
External Examiner	Dr. John Woodward
Internal Examiner(s)	Prof. Michael Madden
	Dr. James McDermott *
Programme Coordinator(s)	Dr. Matthias Nickles, Dr. James McDermott
Instructions	**Answer all 4 questions. All are worth equal marks.**
Duration	2 Hours exam
Number of pages	5 (including this page)
Discipline	Computer Science
Requirements	
Release in Exam Venue	Yes ☒ No ☐
Release to Library	Yes ☒ No ☐
Calculator	Allowed (non-programmable)

Question 1: Basic Python

(a) State the error in this function which is intended to calculate the factorial of input `n`. **[5]**

```
def factorial(n):
    '''return n!, assuming n is integer'''
    if n <= 0:
        print(1)
    else:
        print(n * factorial(n - 1))
```

(b) In the game of tic-tac-toe (also known as noughts and crosses), two players place an `X` or an `O` respectively in cells in a 3x3 grid. Name a Python data structure suitable for storing the grid, and write a snippet of code which would initialise an empty grid and then place an `X` in the bottom-left corner. Use pure Python, no imported libraries. **[5]**

(c) The following code is correct. Explain what `int` does here and why this usage of `int` is correct, including why we do not write `0`, `int()` or `int(0)`. **[5]**

```
# count occurences of unique items in list L, and
# store results in dictionary d.
d = defaultdict(int)
for x in L:
   d[x] += 1
```

(d) Given the following data structure: **[5]**

`T = ('a', ('b', 'c'), ('d', ('e', 'f', ('g', 'h'), ('i', 'j'))))`

Write an expression using `T` that will have the value `('i', 'j')`.

(e) Why is `list` needed here? Why is `map` designed this way? **[5]**

```
L = ['ambulance', 'anteater', 'aardvark']
print(list(map(len, L)))
```

Question 2: Advanced Python

(a) In Python, what are the differences between `math.nan` and `math.inf`? Illustrate with examples of how they arise in numerical code and how they behave in boolean operators such as `>`. **[5]**

(b) Briefly describe the conceptual implementation of the *dictionary* (also known as *hash table*). Refer to table size, `hash`, and the modulus operator `%`. State and explain the time complexity of writing and retrieving items to/from the dictionary. You can ignore the issue of hash collisions. **[10]**

(c) What are the properties of the factorial function which make it suitable for memoisation? **[5]**

(d) What is special about `deque`, the double-ended queue object available in Python `collections`? Explain where this is useful. **[5]**

Question 3: Data Science

(a) The following code fragment uses loops. Rewrite the fragment in a Numpy vectorised style. Assume `L` is a list of `float` values. **[5]**

```
s = 0
for i in range(len(L) - 1):
    if L[i] < L[i+1]:
        s += 1
```

(b) Explain the concept of *batch job* systems in high-performance computing, including advantages and disadvantages. **[5]**

(c) Rewrite the following R code using the `magrittr` pipe, where `D` is a tibble and `a` and `b` are columns. **[5]**

```
select(mutate(D, a=0.4*a, b=0.4*b), year, a, b)
```

(d) The following data is not *tidy*. Explain why not, and show what it would look like in tidy format. **[5]**

Country	Metric	2019	2020
Ireland	Population	5.1	5.2
	GDP	101	102
France	Population	71	72
	GDP	400	410

(e) Suppose we have two dplyr tibbles named `rentals` and `customers`, as shown below. Notice that not every customer ID has an entry in the `customers` table. Write a dplyr join to create a tibble containing all rentals together with the corresponding names and addresses. Names and addresses should be blank wherever they are not available. **[5]**

Rentals table

Date	Movie ID	Customer ID
01-Jan	102	1
02-Jan	101	2
02-Jan	102	3
05-Jan	103	1
05-Jan	104	7

Customer table

Customer ID	Name	Address
1	Bob	11, Haight St
2	Frida	Oxford Circus
3	Carrie	99, Fifth Ave

Question 4: Tools and Applications

The following programs illustrate the legal syntax in a simple programming language called ACIDIC.

```
10 PRINT 1
20 GOTO 10
```

```
1 PRINT 2
2 SET X[0] 5
3 SET X[1] (X[0]+5)
4 GOTO X[0]
```

An ACIDIC program consists of 1 or more lines. Every line consists of a line number and a command, separated by a space. There are several commands: `PRINT`, `SET` (for variable assignment), `IF`, `GOTO` (jump to the given line number).

Every command is followed by one argument (again separated by a space). The argument can involve numerical constants, variables, and operators (e.g. `1`, `10`, `X[0]`, `X[9]`, `+`, `*`, `>`, `==`). The

result of executing a boolean operator such as `==` is an integer 0 or 1. The `SET` command is an exception as it takes *two* arguments: the variable name, and the value to assign to that variable.

Variables `X[0]` up to `X[9]` are the only ones allowed. The `IF` command takes one argument, an expression. If its value True or is a number greater than 0, the next line is executed, otherwise it is not executed. Finally, line numbers need not be successive in the code, as the ACIDIC interpreter will sort them before execution.

(a) Write out a grammar to generate programs in this language. Include all of the commands, constants, variables, and operators mentioned above. Notice that Example 2 above is legal, but will crash, as line 4 tries to `GOTO X[0]`, i.e. to line number 5, which does not exist. Your grammar only has to generate legal programs, and does not have to prevent programs which would crash. **[15]**

(b) Consider the following ACIDIC program. Draw a *graph* representing the flow of the program, where nodes are line numbers. Also, write the adjacency matrix for this graph. **[10]**

```
10 GOTO 30
20 GOTO 40
30 GOTO 20
40 GOTO 30
50 PRINT 'Hello'
```