0.1 Theoretical examination DEV II

The general shape of a theoretical exam for `DEV II` is made up of a series of highly structured open questions.

Question I: abstracting patterns with functions

General shape of the question: *Given a problem description, define one or more functions in order to solve the original problem.*

Concrete example of question: *Define a recursive* `range` *function to create a custom list (only use* `Empty` *and* `Node`*, see Appendix) with all the elements between two given numbers.*

Concrete example of answer: *The resulting code is:*

```
def range(l, u):
  if l > u:
    return Empty()
  else:
    return Node(l, range(l+1,u))
```

Points: *25%.*

Grading: *All points for correct function, minor mistakes (wrong check, some elements might be missing, etc.) half points, wrong function (infinite recursion, iterative version, etc.) zero points.*

Associated learning goals: `FUNABS`, `FUNDEF`, `FUNREC`, `RECDATA`.

Question II: runtime behaviour of functions

General shape of the question: *Given a function definition and a sample call, show stack and heap at all steps of the computation.*

Concrete example of question: *Given the following function definition and a sample call, show stack and heap at all steps of the computation.*

```
def f(n):
  if n <= 1:
    return n
  else:
    return n * f(n-1)
```

```
f(3)
```

Concrete example of answer: *The last call of the stack is :*

```
S: PC   f   PC   n   f   PC   n   f   PC   n
    7  nil   2   3  nil   2   2  nil   2   1
H: always empty
```

The stack will then unwind as follows:

```
S: PC   f   PC   n   f   PC   n   f   PC   n
    7  nil   2   3  nil   2   2   1   3   1
```

```
S: PC   f   PC   n   f   PC   n
    7  nil   2   3  2*1   4   2
```

```
S: PC   f   PC   n
    7  3*2   4   3
```

Points: *25%.*

Grading: *All points for all stack frames and values, half points for at least half correct stack frames and values, otherwise zero points.*

Associated learning goals: `FUNABS`, `FUNDEF`, `FUNREC`, `RECDATA`.

Question III: classes

General shape of the question: *Given a description, give the implementation of a class and its methods in Python.*

Concrete example of question: *Define a `Counter` class with a single method, `Tick`, which increments the internal `cnt` of the class. Also provide an implementation of `__str__`)*

Concrete example of answer: *The resulting code is:*

```
class Counter:
  def __init__(self):
    self.cnt = 0
  def Tick(self):
    self.cnt = self.cnt + 1
  def __str__(self):
    return "Ticked␣" + str(self.cnt) + "␣times"
```

Points: *25%.*

Question IV: standard libraries

General shape of the question: *Define a loop that performs some simple operation on a standard data structure.*

Concrete example of question: *Define a loop that sums all positive elements of a Python list `l` which contains only integers. Print the sum.*

Concrete example of answer: *The resulting code is:*

```
sum = 0
for x in l:
  if x > 0:
    sum = sum + x

print(sum)
```

Points: *25%.*

Grading: *All points for correct answer, otherwise zero points.*

Associated learning goals: `ARR`.