Computational Management Science 1

Fall 2020 Final

registration number: SAMPLE SOLUTION

(Do not write your name on the test - just the 7 digit student id number.)

All examples are evaluated using the Python programming language, version 3.8.

1. (6 points) Writing Code

   (a) (3 points, ≤5 minutes) Functions
   Write a function that takes exactly one non-negative integer as argument and returns the factorial of that number. Use either a loop or recursion to compute the solution. `factorial(0)= 1, factorial(5)=5 * 4 * 3 * 2 * 1 = 120`.

```python
def factorial(n: int) -> int:
    """
    Return the factorial of ‘n‘.

    >>> factorial(5)
    120
    """
    return 1 if n <= 1 else n * factorial(n - 1)
```

   (b) (3 points, ≤5 minutes) Classes and data structures
   Implement a simple `Student` data structure in Python. The data structure must be capable of storing the data of a student in this course: `first_name`, `last_name`, `id_number`, `scores`. The first three are passed when constructing an instance while `scores` must be initialized to an empty list.
   Write a **minimalistic** class (`__init__(.)`). You don't need to implement any functionality, just a class that stores the required data. Note that your implementation must be self-contained and must not rely on any third party functionality not provided by you. Don't forget to write docstrings in order to receive full points.

```python
class Student:
    """A student participating in a course."""
    def __init__(self, first_name: str, last_name: str, id_number: str):
        self.first_name = first_name
        self.last_name = last_name
        self.id_number = id_number
        self.scores = []
```

2. (6 points, ≤10 minutes) Relational Database Design

Design a relational database to store the following information:

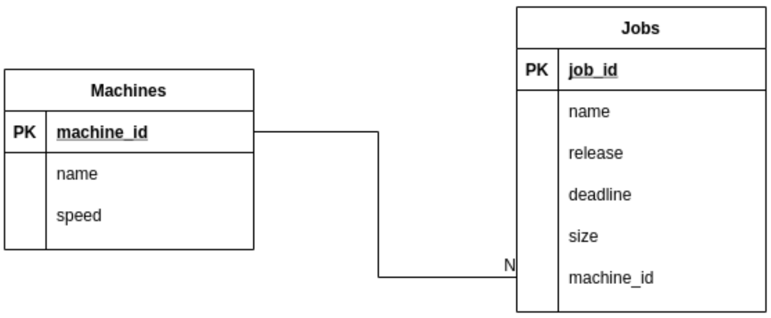
We want to store a list of machines for which we know the name and the speed. In addition, we want to store a list of jobs for which we know a name, release time, deadline and size (integer).

(a) Every job is assigned to a unique machine on which it has to be produced.

In addition, please also give an example on how to store the case of 2 machines and 3 jobs, the first two being scheduled on the first machine, the third scheduled on the 2nd machine.

(b) Every job can be assigned on none, one or multiple machines.

In addition, please also give an example with two machines and two jobs. The first job should be assigned to the 2nd machine and the 2nd job should be assigned to both machines.

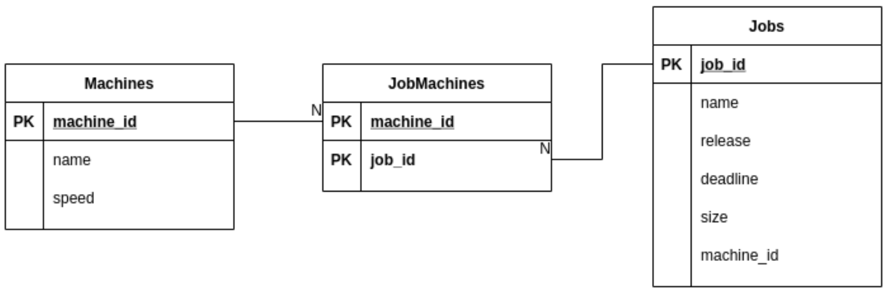


Machines

id	name	speed
1	Alpha	1.5
2	Kappa	3

Jobs

id	name	release	deadline	size	machine_id
1	rings-y	2021-01-25 17:10:00	2021-01-25 19:10:30	250	1
2	rings-b	2021-02-25 13:15:00	2021-01-26 12:00:00	1025	1
3	squares-b	2021-02-26 04:00:00	2021-01-26 15:30:00	350	2



Machines

id	name	speed
1	Alpha	1.5
2	Kappa	3

Jobs

id	name	release	deadline	size	machine_id
1	rings-y	2021-01-25 17:10:00	2021-01-25 19:10:30	250	
2	rings-b	2021-02-25 13:15:00	2021-01-26 12:00:00	1025	

JobMachines

job_id	machine_id
1	2
2	1
2	2

3. (9 points, ≤10 minutes) Libraries

   (a) (1 points)
   What is the purpose of a named tuple? Which library provides this data structure?
   A named tuple is a tuple that allows accessing attributes by name like a dictionary and by named attribute access in addition to position. It is part of the Python standard library (in the `collections` module).

   (b) (3 points)
   What is the most important category of automated tests. Is their execution generally fast or slow? When should they be executed?
   Unit tests are the foundation of any decent automated test strategy. They are generally very fast and should be executed before and after any code change. Ideally, they are run by the CI as well.

   (c) (3 points)
   What is the purpose of doctests? Provide one example of a function with a doctest (use valid syntax).
   A doctest shows how a code unit is supposed to be used with a general example.

```python
def factorial(n: int) -> int:
    """
    Return the factorial of ‘n‘.

    >>> factorial(5)
    120
    """
    return 1 if n <= 1 else n * factorial(n - 1)
```

   (d) Finally, name one thing you like about this course and one thing that should be improved in the future (be honest!) (2p).
   UP TO THE STUDENTS...

4. (12 points, ≤10 minutes) Reading and Understanding Code
What is the output of the following code snippets? Write exactly what the output of each snippet is if the snippet is the sole content of a Python file. If the output is an error message, it is enough to write "ERROR". If there is no output, write "-"

(a) Loop

```
n = 10
d = 6
work = True
while work:
    if n / d == 0:
        print(d)
        work = False
    d -= 1
```

```
ERROR
```

(b) Simple calculation

```
a = 2
b = 3
c = a * b**2 + 2 * b - 1
print(c)
```

```
23
```

(c) Function

```
def funnyswap(a, b):
    c = a
    a = b
    b = c
    return a, b

a = 5
b = 10
c, d = funnyswap(a, b)
print(a, b, c, d)
```

```
5 10 10 5
```

(d) Lists and tuples

```
values = [-5, 4, 3]
values[0] = 5
```

-

(e) String operations

```
s = "Everyone wants to get the grade 1 in CMS."
print(s[3])
print(s[:5] + ' ' + s[-4:-1])
```

```
r
Every CMS
```

(f) List comprehension

```
integers = [2*i for i in range(5)]
print(max(integers))
```

```
8
```

5. (9 points, ≤10 minutes)

   (a) (3 points)
   What is a zero-sum game? In addition to the general definition, give one example of a zero-sum game and one example of a game that is not a zero-sum game.

   A zero-sum game is a game in which the sum of the gain of all players that gain something minus the loss of all players that loose something equals 0.

   Example of a zero-sum-game: Chess where the loser has to pay the winner 1 EUR and no money is exchanged when there is a draw.

   Example of a non-zero-sum game: Any competitive casino betting game, in which some part of the bets goes to the casino (assuming the casino is not considered a player).

   (b) (3 points)
   What is recursion? What alternative programming concept can you always use instead of recursion? Write a short code example of a recursive function.

   You can always use loops instead of recursion.

```
def binary_search(sorted_list, l, r, key):
    if r >= l:
        m = (l + r) // 2
        if sorted_list[m] == key:
            return m
        elif sorted_list[m] > key:
            return binary_search(sorted_list, l, m-1, key)
        elif
            return binary_search(sorted_list, m+1, r, key)
    else:
        return -1
```

   (c) (3 points)
   What is polymorphism? Give one example where and how you would use polymorphism.

   Objects of a subclass can be used where an object of the base class is expected. If a method is called the implementation in the subclass is executed.

   Use case: Different enemy types in a game can all move, be attacked, attack, . . .

   How used: The base class a move method that contains the basic implementation of movement in the game. In the subclass the move method for instance contains some extra effects that this enemy has while moving, like destroying the ground below it and then calls in addition the move method of the base class.

6. (6 points, ≤5 minutes) Student Activities
A fictitious university offers a very large number of different spare time activities. Each student is allowed to enroll in exactly one activity or may choose not to enroll in any. The data is currently stored in a Python dictionary which uses student ids as keys and the name of the student's activity as value. If a student picked no activity, None is used. In order to know who is enrolled in a given activity, the dictionary should be transformed into another dictionary with activities as keys and a list of enrolled students as values. The sort order of students in the lists is not relevant. Write a function `transform(before)` that returns a new dictionary in the desired format. For example

```
before = {'9825756': 'archery',
          '0315423': 'table tennis',
          '1303451': 'archery',
          '0834154': None}
after = {'archery': ['9825756', '1303451'],
         'table tennis': ['0315423'],
         None: ['0834154']}
```

```
def transform(by_ids: dict) -> dict:
    """
    Return a new dict with the argument's keys and values flipped.

    The output's values are lists with the input's keys.

    >>> transform({'a': 'b'})
    {'b': ['a']}
    """
    by_activity = {}
    for key, value in by_ids.items():
        by_activity[value] = by_activity.get(value, []) + [key]
    return by_activity
```