LAB EXAM (29 March 2022 : 1400-1550)

Before starting

- The exam runs 1400-1550.
- Answer all questions.
- Upload all code to Einstein.
- All lab exam rules apply.

Question 1 [10 marks]

- In module `student_v1_121.py` define a `Student` class to model a leaving certificate student.
- A student has an associated name and CAO number.
- When your class is correctly implemented running the following program should produce the given output.

```python
from student_v1_121 import Student

def main():
    s1 = Student('Boris Spassky', 21345654)
    s2 = Student('Bobby Fischer', 21907321)

    assert(s1.name == 'Boris Spassky')
    assert(s1.cao == 21345654)

    print(s1)
    print(s2)

if __name__ == '__main__':
    main()
```

```
Name: Boris Spassky
CAO: 21345654
Name: Bobby Fischer
CAO: 21907321
```

Question 2 [10 marks]

Higher Level Grade	Points	Ordinary Level Grade	Points
H1	100		
H2	88		
H3	77		
H4	66		
H5	56	O1	56
H6	46	O2	46
H7	37	O3	37
H8	0	O4	28

Higher Level Grade	Points	Ordinary Level Grade	Points
		O5	20
		O6	12
		O7	0
		O8	0

- In module *student_v2_121.py* extend the `Student` class to support the recording and retrieval of per-subject grades for a student.
- See the table above for a list of awardable grades.
- You can assume all awarded grades will be drawn from those listed in the table.
- The collection of subjects for which a student may register is arbitrary.
- You can assume the name of each subject consists entirely of lowercase letters.
- When your class is correctly implemented running the following program should produce the given output.

```python
from student_v2_121 import Student

def main():

    s1 = Student('Boris Spassky', 17345654)
    s2 = Student('Bobby Fischer', 17907321)

    s1.add_grade('english', 'H3')
    s1.add_grade('irish', 'O2')
    print(s1.get_grade('english'))

    s2.add_grade('english', 'H4')
    s2.add_grade('irish', 'H6')
    s2.add_grade('chemistry', 'O5')
    print(s2.get_grade('physics'))

if __name__ == '__main__':
    main()
```

```
H3
N/A
```

Question 3 [15 marks]

- In module *student_v3_121.py* extend the `Student` class to report a student's CAO points.
- A student's CAO points are calculated by summing the points awarded for their three top-scoring subjects.
- Where a student has three or fewer recorded grades their CAO points is the sum of all their points.
- See the table above for the points awarded for each grade.
- When your class is correctly implemented running the following program should produce the given output.

```python
from student_v3_121 import Student

def main():
    s1 = Student('Boris Spassky', 21345654)
    s2 = Student('Bobby Fischer', 21907321)

    s1.add_grade('english', 'H2')
    s1.add_grade('irish', 'O4')
    s1.add_grade('french', 'H3')
    s1.add_grade('physics', 'H3')
    print(s1)

    print(s2)

if __name__ == '__main__':
    main()
```

```
Name: Boris Spassky
CAO: 21345654
Points: 242
Name: Bobby Fischer
CAO: 21907321
Points: 0
```

Question 4 [10 marks]

- In module *student_v4_121.py* extend the `Student` class to support the comparison of students.
- Comparison is carried out in terms of a student's CAO points.
- When your class is correctly implemented running the following program should produce the given output.

```
from student_v4_121 import Student

def main():

    s1 = Student('Boris Spassky', 21345654)
    s2 = Student('Bobby Fischer', 21907321)
    s3 = Student('Mikhail Tal', 21884786)

    s1.add_grade('english', 'H2')
    s1.add_grade('irish', 'H3')
    s1.add_grade('chemistry', 'H5')
    print(s1)

    s2.add_grade('irish', 'H3')
    s2.add_grade('physics', 'H2')
    s2.add_grade('french', 'O1')
    print(s2)

    s3.add_grade('art', 'H1')
    s3.add_grade('music', 'H2')
    s3.add_grade('woodwork', 'H2')
    print(s3)

    assert(s1 == s2)
    assert(s1 < s3)
    assert(s3 > s2)

if __name__ == '__main__':
    main()
```

```
Name: Boris Spassky
CAO: 21345654
Points: 221
Name: Bobby Fischer
CAO: 21907321
Points: 221
Name: Mikhail Tal
CAO: 21884786
Points: 276
```

Question 5 [10 marks]

- In module *classlist_v1_121.py* define a `Classlist` class to model a collection of leaving certificate students.
- A classlist is essentially a mapping from student CAO numbers to `Student` objects.
- **You must include in classlist_v1_121.py a copy of your Student class definition from student_v1_121.py.**
- Students can be added to and removed from the classlist using the `add()` and `remove()` methods respectively.
- Removing a student who is not in the classlist has no effect.
- A `lookup()` method returns a `Student` object if a given student is in the classlist and `None` otherwise.
- When your class is correctly implemented, running the following program should produce no output.

```
from classlist_v1_121 import Student, Classlist

def main():

    cl = Classlist()
    s1 = Student('Boris Spassky', 21345654)
    s2 = Student('Bobby Fischer', 21907321)

    cl.add(s1)
    cl.add(s2)

    s = cl.lookup(21345654)
    assert(isinstance(s, Student))
    assert(s.name == 'Boris Spassky')
    assert(s.cao == 21345654)

    cl.remove(21345654)
    s = cl.lookup(21345654)
    assert(s is None)

if __name__ == '__main__':
    main()
```

Question 6 [10 marks]

- In module *classlist_v2_121.py* extend the `Classlist` class to support the printing of a classlist.
- Printing a classlist prints all student details in ascending order of their CAO numbers.

- **You must include in classlist_v2_121.py a copy of your Student class definition from student_v1_121.py.**
- When your class is correctly implemented, running the following program should produce the given output.

```python
from classlist_v2_121 import Student, Classlist

def main():

    cl = Classlist()
    s1 = Student('Boris Spassky', 21345654)
    s2 = Student('Bobby Fischer', 21907321)
    s3 = Student('Mikhail Tal', 21884786)

    cl.add(s1)
    cl.add(s2)
    cl.add(s3)

    print(cl)

if __name__ == '__main__':
    main()
```

```
Name: Boris Spassky
CAO: 21345654
Name: Mikhail Tal
CAO: 21884786
Name: Bobby Fischer
CAO: 21907321
```

Question 7 [15 marks]

- In module `classlist_v3_121.py` extend the `Classlist` class to support retrieval of the most popular subject amongst students.
- The most popular subject is the one for which the most students are registered.
- You can assume all students are registered for at least one subject and that the most popular one is unique i.e. there will be no ties.
- **You must include in classlist_v3_121.py a copy of your Student class definition from student_v2_121.py.**
- When your class is correctly implemented running the following program should produce the given output.

```python
from classlist_v3_121 import Student, Classlist

def main():

    cl = Classlist()
    s1 = Student('Boris Spassky', 21345654)
    s2 = Student('Bobby Fischer', 21907321)
    s3 = Student('Mikhail Tal', 21884786)

    s1.add_grade('english', 'H1')
    s1.add_grade('irish', 'O4')

    s2.add_grade('english', 'H2')
    s2.add_grade('french', '05')
    s2.add_grade('spanish', '01')

    s3.add_grade('english', '03')
    s3.add_grade('irish', '03')

    cl.add(s1)
    cl.add(s2)
    cl.add(s3)

    print(cl.most_popular_subject())

if __name__ == '__main__':
    main()
```

```
english
```

Question 8 [20 marks]

- A pattern of length P consists of L lowercase letters and D dashes.
- P is a positive integer where P = L + D and 0 <= L <= P and 0 <= D <= P.
- Write a program called `pattern_121.py` that reads a pattern from `stdin` followed by a list of words (one per line) from `stdin`.
- Each word read from `stdin` consists entirely of lowercase letters.
- Your program should output a comma-separated list of all words read from `stdin` that match the supplied pattern.
- If no words match the pattern your program should output nothing.
- Matching words should be output in the order they are read from `stdin`.
- Matching rules:
  - Words can only match patterns of the same length, and,
  - a dash can match any letter, and,
  - a letter can only match itself.
- In the example below *sprat* and *splat* match the pattern *sp-at*:

```
$ cat pattern_stdin_00_121.txt
sp-at
scrape
sprat
stray
splat
sprats
```

```
$ python3 pattern_121.py < pattern_stdin_00_121.txt
sprat, splat
```

- In the example below *scrape* matches the pattern *sc-ap-*:

```
$ cat pattern_stdin_01_121.txt
sc-ap-
scrape
```

```
sprat
stray
splat
sprats
```

```
$ python3 pattern_121.py < pattern_stdin_01_121.txt
scrape
```