R00182510 DA Assignment 1

Both Task 1 and Task 2 were implemented in a single file R00182510_DA_A1_Code.py

To execute Task_1, in the main() function please assign the value ‘Task_1’ to the execute variable.

To execute Task_2, in the main() function please assign the value ‘Task_2’ to the execute variable.

Task 1

In this task a constraint satisfaction model was developed that solves a logical puzzle as posted in the Assignment1: Constraint programming document.

Puzzle -> Carol, Elisa, Oliver and Lucas are going to university. Find the exact university.

Subtask A:

Objects:

```
students = ["Student #1", "Student #2", "Student #3", "Student #4"]
```

Predicates and attributes:

```
names = ["Carol", "Elisa", "Oliver", "Lucas"]
universities = ["London", "Cambridge", "Oxford", "Edinburgh"]
genders = ["boy", "girl"]
nationalities = ["Australia", "USA", "South Africa", "Canada"]
majors = ["History", "Medicine", "Law", "Architecture"]
```

Decision Variables:

In function Task1() of the code, Lines 58 to 93 shows the creation of below listed decision variables based on the predicates and attributes shown above.

```
student_name, student_university, student_gender, student_nationality,
student_major
```

Subtask B:

Explicit Sentence Constraint - 1

One of them is going to London implies the below conditions.

In function Task1() of the code, Lines 193 to 197 implements this constraint

```
# University(s1, London) -> !University(s2,London) & !University(s3, London) & -
!University(s4, London)
# University(s2, London) -> !University(s1,London) & !University(s3, London) &
!University(s4, London)
```

```
# University(s3, London) -> !University(s1,London) & !University(s2, London) &
!University(s4, London)
# University(s34, London) -> !University(s1,London) & !University(s2, London) &
!University(s3, London)
```

<u>Explicit Sentence Constraint - 2</u>

Exactly one boy and one girl chose a university in a city with the same initial of their names implies the below conditions.

In function Task1() of the code, Lines 210 to 260 implements this constraint

```
# Gender(student, 'girl'), Name(student, 'Carol'), University(student, Cambridge)
-> !University(other, 'Edinburgh'), Gender(other, 'girl'), Name(other, 'Elisa')

# Gender(student, 'girl'), Name(student, 'Elisa'), University(student, Edinburgh)
-> !University(other, 'Cambridge'), Gender(other, 'girl'), Name(other, 'Carol')

# Gender(student, 'boy'), Name(student, 'Lucas'), University (student, London) ->
!University(other, 'Oxford'), Gender(other, 'boy'), Name(other, 'Oliver')

# Gender(student, 'boy'), Name(student, 'Oliver'), University(student, Oxford) ->
!University(other, 'London'), Gender(other, 'boy'), Name(other, 'Lucas')
```

<u>Explicit Sentence Constraint - 3</u>

A boy is from Australia, the other studies History implies the below conditions

In function Task1() of the code, Lines 269 to 276 implements this constraint

```
# Gender(student, 'girl') ->! Nationality(student, 'Australia')
# Gender(student, 'girl') -> !Major(student, 'History')
# Gender(student, 'boy') -> !Major(student, 'History') OR !Nationality(student,
                                                                'Australia'))
```

<u>Explicit Sentence Constraint - 4</u>

A girl goes to Cambridge, the other studies Medicine implies the below conditions

In function Task1() of the code, Lines 284 to 290 implements this constraint

```
# Gender(student, 'boy') -> !University(student, 'Cambridge')
# Gender(student, 'boy') -> !Major(student, 'Medicine')
# Gender(student, 'girl') -> !University(other, 'Cambridge') OR !Major(other,
                                                                 'Medicine')
```

<u>Explicit Sentence Constraint - 5</u>

Oliver studies Law or is from USA; He is not from South Africa, implies the below conditions

In function Task1() of the code, Lines 296 to 300 implements this constraint

```
# Name(student, 'Oliver') -> !Nationality(student, 'South Africa')
# Name(student, 'Oliver') -> Major(student, 'Law') OR Nationality(student, 'USA')
```

Explicit Sentence Constraint - 6

The student from Canada is a historian or will go to Oxford, implies the below conditions

In function Task1() of the code, Lines 306 to 308 implements this constraint

```
# Nationality(student, 'Canada') -> Major(student, 'Historian') OR
                                                   University(student, 'Oxford')
```

Explicit Sentence Constraint - 7

The student from South Africa is going to Edinburgh or will study Law, implies the below conditions

In function Task1() of the code, Lines 314 to 317 implements this constraint

```
# Nationality(student, 'South Africa') -> Major(student, 'Law') OR
University(student, 'Edinburgh')
```

Subtask C:

Implicit Sentence constraint – 1

Every student has at least has one name, one university, one nationality, one major and one gender.

In function Task1() of the code, Lines 104 to 128 implements this constraint

Implicit Sentence constraint – 2

Every student has no more than one name, one university, one nationality, one major and one gender.

In function Task1() of the code, Lines 136 to 153 implements this constraint

Implicit Sentence constraint – 3

Every student has a different name, university, nationality and major

In function Task1() of the code, Lines 160 to 173 implements this constraint

Implicit Sentence constraint – 4

There are two boys (Oliver, Lucas) and two girls (Elisa, Carol)

In function Task1() of the code, Lines 179 to 183 implements this constraint

```
# Gender(student, girl) -> Name('Elisa') or Name('Carol')
# Gender(student, boy) -> Name('Oliver') or Name('Lucas')
```

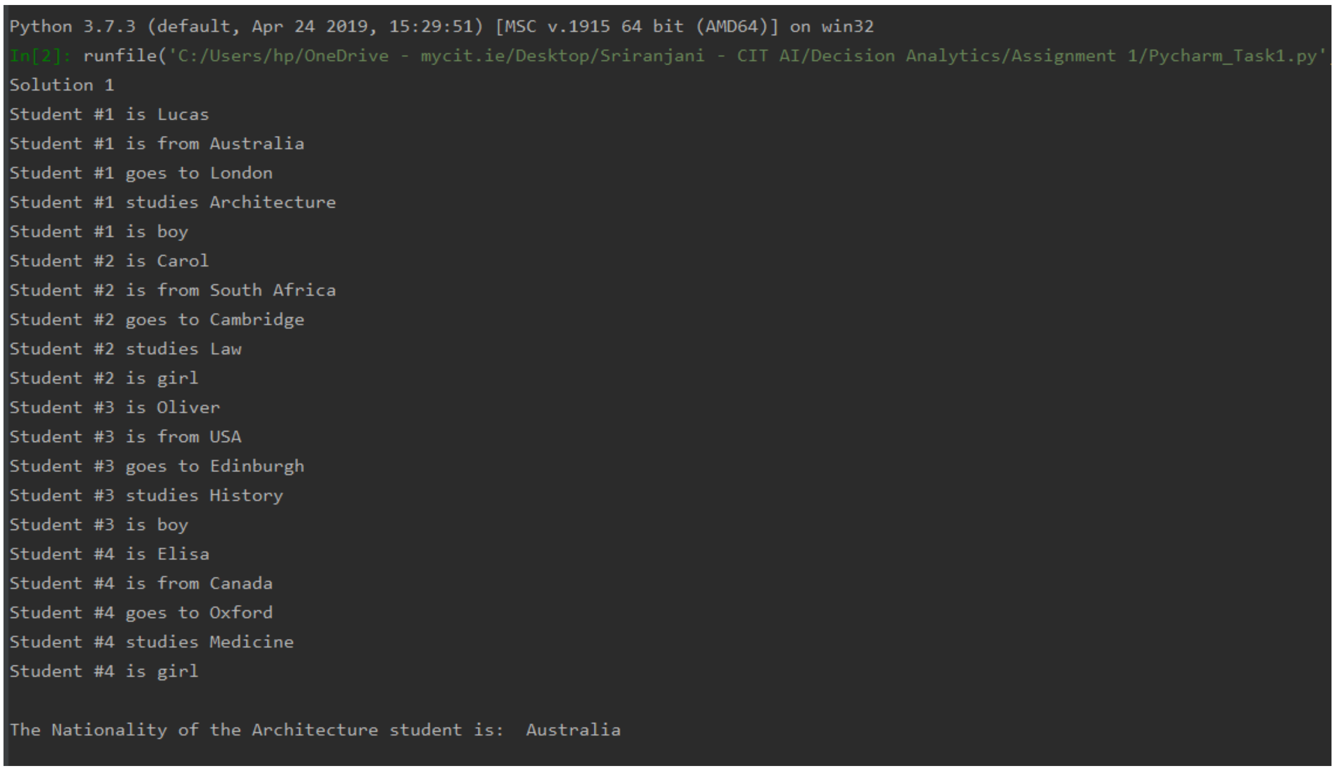
Implicit Sentence constraint – 5

Received multiple solutions with the same assignment of values for all variables but with different ordering of students(duplicates). To overcome this and receive one unique solution, the student number is assigned the names in one order.

In function Task1() of the code, Lines 323 to 326 implements this constraint

Subtask D:

The class Task1_SolutionPrinter(), prints the solution based on the above model constraints. The solution for this task is shown below.



```
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] on win32
In[2]: runfile('C:/Users/hp/OneDrive - mycit.ie/Desktop/Sriranjani - CIT AI/Decision Analytics/Assignment 1/Pycharm_Task1.py'
Solution 1
Student #1 is Lucas
Student #1 is from Australia
Student #1 goes to London
Student #1 studies Architecture
Student #1 is boy
Student #2 is Carol
Student #2 is from South Africa
Student #2 goes to Cambridge
Student #2 studies Law
Student #2 is girl
Student #3 is Oliver
Student #3 is from USA
Student #3 goes to Edinburgh
Student #3 studies History
Student #3 is boy
Student #4 is Elisa
Student #4 is from Canada
Student #4 goes to Oxford
Student #4 studies Medicine
Student #4 is girl

The Nationality of the Architecture student is:  Australia
```

Subtask E:

Evaluation of each explicit constraint/sentence to see if its redundant.

1) **One of them is going to London**
   This sentence/constraint is redundant as the implicit constraints 1 and 2 given above ensure the one of the students goes to London. Hence this sentence can be omitted.

2) **Exactly one boy and one girl chose a university in a city with the same initial of their names**

This is a mandatory constraint to determine a solution to the puzzle. This sentence does not create redundancy and hence cannot be omitted.

3) **A boy is from Australia, the other studies History**
This is a mandatory constraint to determine a solution to the puzzle. This sentence does not create redundancy and hence cannot be omitted.

4) **A girl goes to Cambridge, the other studies Medicine**
This is a mandatory constraint to determine a solution to the puzzle. This sentence does not create redundancy and hence cannot be omitted.

5) **Oliver studies Law or is from USA; He is not from South Africa**
This is a mandatory constraint to determine a solution to the puzzle. This sentence does not create redundancy and hence cannot be omitted.

6) **The student from Canada is a historian or will go to Oxford**
This is a mandatory constraint to determine a solution to the puzzle. This sentence does not create redundancy and hence cannot be omitted.

7) **The student from South Africa is going to Edinburgh or will study Law**
This is a mandatory constraint to determine a solution to the puzzle. This sentence does not create redundancy and hence cannot be omitted.

Task 2

The goal is to determine which projects can be delivered and what subcontractors should be contracted while making sure that the overall profit margin is at least €2500.

Subtask A:

In function Task2() of the code, Lines 399 to 439 implements this subtask to extract all the relevant data from the input excel sheet.

The following evaluations were performed to ensure only the valid combination of values were used for creation of decision variables in Subtask B.

Valid months for each project

{'Project A': ['M1', 'M2', 'M3'], 'Project B': ['M3', 'M4', 'M5'], 'Project C': ['M4', 'M5', 'M6', 'M7', 'M8'], 'Project D': ['M2', 'M3', 'M4', 'M5'], 'Project E': ['M8', 'M9'], 'Project F': ['M9', 'M10', 'M11'], 'Project G': ['M5', 'M6', 'M7'], 'Project H': ['M8', 'M9', 'M10', 'M11'], 'Project I': ['M10', 'M11', 'M12']}

Valid jobs for each project month combination

{('Project A', 'M1'): ['Job A'], ('Project A', 'M2'): ['Job B'], ('Project A', 'M3'): ['Job C'], ('Project B', 'M3'): ['Job G'], ('Project B', 'M4'): ['Job K'], ('Project B', 'M5'): ['Job M'], ('Project C', 'M4'): ['Job H'], ('Project C', 'M5'): ['Job E'], ('Project C', 'M6'): ['Job G'], ('Project C', 'M7'): ['Job B'], ('Project C', 'M8'): ['Job E'], ('Project D', 'M2'): ['Job D'], ('Project D', 'M3'): ['Job F'], ('Project D', 'M4'): ['Job I'], ('Project

D', 'M5'): ['Job H'], ('Project E', 'M8'): ['Job J'], ('Project E', 'M9'): ['Job A'], ('Project F', 'M9'): ['Job B'], ('Project F', 'M10'): ['Job C'], ('Project F', 'M11'): ['Job K'], ('Project G', 'M5'): ['Job L'], ('Project G', 'M6'): ['Job M'], ('Project G', 'M7'): ['Job E'], ('Project H', 'M8'): ['Job A'], ('Project H', 'M9'): ['Job B'], ('Project H', 'M10'): ['Job D'], ('Project H', 'M11'): ['Job I'], ('Project I', 'M10'): ['Job L'], ('Project I', 'M11'): ['Job F'], ('Project I', 'M12'): ['Job K']}

<u>Valid contractors that can be assigned for project, month and job combination</u>

{('Project A', 'M1', 'Job A'): ['Contractor A', 'Contractor H'], ('Project A', 'M2', 'Job B'): ['Contractor E', 'Contractor J'], ('Project A', 'M3', 'Job C'): ['Contractor F', 'Contractor K'], ('Project B', 'M3', 'Job G'): ['Contractor C', 'Contractor G', 'Contractor I'], ('Project B', 'M4', 'Job K'): ['Contractor B', 'Contractor K'], ('Project B', 'M5', 'Job M'): ['Contractor B', 'Contractor K'], ('Project C', 'M4', 'Job H'): ['Contractor H', 'Contractor K'], ('Project C', 'M5', 'Job E'): ['Contractor A', 'Contractor E'], ('Project C', 'M6', 'Job G'): ['Contractor C', 'Contractor G', 'Contractor I'], ('Project C', 'M7', 'Job B'): ['Contractor E', 'Contractor J'], ('Project C', 'M8', 'Job E'): ['Contractor A', 'Contractor E'], ('Project D', 'M2', 'Job D'): ['Contractor D', 'Contractor H'], ('Project D', 'M3', 'Job F'): ['Contractor F', 'Contractor J'], ('Project D', 'M4', 'Job I'): ['Contractor A', 'Contractor G'], ('Project D', 'M5', 'Job H'): ['Contractor H', 'Contractor K'], ('Project E', 'M8', 'Job J'): ['Contractor C', 'Contractor J'], ('Project E', 'M9', 'Job A'): ['Contractor A', 'Contractor H'], ('Project F', 'M9', 'Job B'): ['Contractor E', 'Contractor J'], ('Project F', 'M10', 'Job C'): ['Contractor F', 'Contractor K'], ('Project F', 'M11', 'Job K'): ['Contractor B', 'Contractor K'], ('Project G', 'M5', 'Job L'): ['Contractor A', 'Contractor D'], ('Project G', 'M6', 'Job M'): ['Contractor B', 'Contractor K'], ('Project G', 'M7', 'Job E'): ['Contractor A', 'Contractor E'], ('Project H', 'M8', 'Job A'): ['Contractor A', 'Contractor H'], ('Project H', 'M9', 'Job B'): ['Contractor E', 'Contractor J'], ('Project H', 'M10', 'Job D'): ['Contractor D', 'Contractor H'], ('Project H', 'M11', 'Job I'): ['Contractor A', 'Contractor G'], ('Project I', 'M10', 'Job L'): ['Contractor A', 'Contractor D'], ('Project I', 'M11', 'Job F'): ['Contractor F', 'Contractor J'], ('Project I', 'M12', 'Job K'): ['Contractor B', 'Contractor K']}

Subtask B:

The decision variable to identify the projects to take on is created in function Task2() of the code, lines 443 to 445.

The decision variables to decide which contractor, is working on which project and when is created in function Task2() of the code, lines 450 to 458.

Decision variables are created only for the valid combination of (project, month, job, contractor) determined from Subtask A.

A constraint to ensure that projects do not run all months is implemented in function Task2(), lines 462 to 468.

Subtask C:

A constraint that a contractor cannot work on two projects simultaneously.

In function Task2(), lines 471 to 480 implements this constraint

Subtask D and E:

Both the subtasks were coupled together a they required the same set of variables involved.

Subtask D - A constraint that if a project is accepted to be delivered then exactly one contractor per job of the project needs to work on it.

Subtask E - A constraint that if a project is not taken on then no one should be contracted to work on it.

In function Task2(), lines 485 to 500 implements both the above constraints

Subtask F:

Implementing the project dependency constraints

In function Task2(), lines 503 to 513 implements this constraint

Subtask G:

Implementing the constraint that the profit margin is at least €2500

In function Task2(), lines 517 to 532 implements this constraint

Subtask H:

The model is solved with the above constraints and the solutions are printed using the class Task2_SolutionPrinter(). This is implemented in lines 344 to 392.

The model resulted in 43 solutions as shown in the attached file R00182510_DA_Task2_Solutions.

In all the solutions the projects taken remained constant as shown below.

Projects Taken: ['Project A', 'Project B', 'Project C', 'Project D', 'Project E', 'Project G', 'Project H', 'Project I']

However, the contractor assignments differ with each solution, and thereby satisfying all the constraints and having a profit margin >= 2500.