DECISION ANALYTICS
ASSIGNMENT 1

Name : Pavithra Ramasubramanian Ramachandran
Student ID : R00183771

Task 1:

Code File : da1.py

Note:
- The code is commented with sections Task A, B,.....E and the constraint that is being implemented.

Task A:

Objects : Carol, Elisa, Oliver and Lucas

Attributes :
- university (London, Cambridge, Oxford, Edinburg)
- nationality (Australia, USA, South Africa, Canada)
- course (History, Medicine, Law, Architecture)
- gender (male, female)

Boolean Decision variables/Predicates:
- 4X4 variables for name – university combination
- 4X4 variables for name – nationality combination
- 4X4 variables for name – course combination
- 4X2 variables for name – gender combination

Task B:

Explicit Constraints:

1. **One of them is going to London**
   - First Order Logic - **∀x : university(x,London)**
   - This constraint is included in the implicit constraint 3. Therefore, explicit implementation of this constarint is not necessary.

2. **Exactly one boy and one girl chose a university in a city with the same initial of their names.**
   - First Order Logic -
     1. **((university(Carol,Cambridge) v university(Elisa, Edinburg))**
        ^ ¬((university(Carol,Cambridge) ^ university(Elisa, Edinburg))
     2. **((university(Oliver,Oxford) v university(Lucas, London))**
        ^ ¬((university(Oliver,Oxford) ^ university(Lucas, London))

   - This constraint is implemented using the XOR operator in CP-SAT model.
   - It indicates that either Carol goes to Cambridge or Elisa goes to Edinburg. And, either Oliver goes to Oxford or Lucas goes to London.

3. **A boy is from Australia, the other studies History**

- First Order Logic -
  1. **∀x : nationality(x,Australia) ^ ¬course(x,History)=> gender(x,male)**
  2. **∀x : course(x,History) => gender(x,male)**
- To implement this constraint, two ‘OnlyEnforceIf’s are used where we add a constraint to the model stating that one of the objects can be from ‘Australia’ only if the object is a ‘male’ and only if the object does not study ‘History’
- Explicit mentioning of a constraint where the other ‘male’ studies ‘ History’ does not give an output at all although in the final solution the other boy is assigned ‘History’ as the course.

4. **A girl goes to Cambridge, the other studies Medicine**

- First Order Logic -
  1. **∀x : university(x,Cambridge) ^ ¬course(x,Medicine)=> gender(x,female)**
  2. **∀x : course(x,Medicine) => gender(x,female)**
- To implement this constraint, two ‘OnlyEnforceIf’s are used where we add a constraint to the model stating that one of the objects can go to ‘Cambridge’ only if the object is a ‘female’ and only if the object does not study ‘Medicine’
- Explicit mentioning of a constraint where the other ‘female’ studies ‘ Medicine’ does not give an output at all although in the final solution the other girl is assigned ‘Medicine’ as the course.

5. **Oliver studies Law or is from USA; He is not from South Africa**

- First Order Logic -
  1. **course(Oliver,Law) v nationality(Oliver, USA)**
  2. **¬nationality(Oliver, South Africa)**
- An XOR operator is used to indicate that Oliver either studies ‘Law’ or is from the ‘USA’.
- And a Not() Operator is used to indicate that Oliver is not from ‘ South Africa’.

6. **The student from Canada is a historian or will go to Oxford.**

- First Order Logic -
  ∀x : (course(x,History) v university(x,Oxford)) => nationality(Canada)
- The most suitable operator for this scenario is XOR to indicate that the student from ‘Canada’ is wither a ‘Historian’ or will go to ‘Oxford’. But we cannot use an XOR gate as it does not support the ‘OnlyEnforceIf’ statement.
- Therefore, an OR operator is used to add this constraint, where there are three possiblities
  1. The object will be a Historian and from Canada
  2. The object will go to Oxford and will be from Canada
  3. The object will be a Historian, will go to Oxford and will be from Canada.
- Although the 1-1(Historian and Oxford) solution is possible, due to the other constraints, this 1-1(3) will get nullified and only (1 or 2) will occur.

7. T**he student from South Africa is going to Edinburgh or will study Law.**

First Order Logic -

∀x : (course(x,Law) v university(x,Edinburg)) => nationality(South Africa)

- This constraint is also similar to the 6th constraint and is solved in a similar manner.

Task C:

Implicit constraints:

1. Carol and Elisa are females. Oliver and Lucas are males.

2. Every student goes to a different university, is from a different nationality and studies a different course.
3. Every student should have a property(nationality, course, university, gender)
4.Every student should posses a maximum of one property(university, nationality and course and gender)

Task D:

On solving the CP-SAT problem using the CP-SAT solver, we obtain the following solution:

solution 1
- Carol:
  - Cambridge
  - South Africa
  - Law
  - female
- Elisa:
  - Oxford
  - Canada
  - Medicine
  - female
- Oliver:
  - Edinburg
  - USA
  - History
  - male
- Lucas:
  - London
  - Australia
  - Architecture
  - male

This indicates that **The nationality of the Architecture student is ‘ Australian’.**

Task E :

Constraint No.	Constraint	Redundant(Yes/ No)	Explanation
1	One of them is going to London	Yes	Explicit mention of the constraint is not necessary as the contsraint by itself is implicit.
2	Exactly one boy and one girl chose a university in a city with the same initial of their names	No	This constraint is necessary as it gives 10 solutions otherwise.
3	A boy is from Australia, the other studies History	No	This constraint is necessary as it gives 13 solutions otherwise.

			However, there is not need to explicitly mention that the other boy studies ‘History’.
4	A girl goes to Cambridge, the other studies Medicine	No	This constraint is necessary as it give 14 solutions otherwise. However, there is not need to explicitly mention that the other girl studies ‘Medicine’.
5	Oliver studies Law or is from USA; He is not from South Africa	Yes and No (He is not from South Africa – this constraint need not be implemented expicitly)	On ommission of the first half - ‘Oliver studies Law or is from USA’, we get about 6 solutions. And on ommission of the entire constraint we get about 8 solutions. However, on ommision of the latter half of the constarint, we get one solution which is correct.
6	The student from Canada is a historian or will go to Oxford	No	We get 4 solutions if we remove this constraint.
7	The student from South Africa is going to Edinburgh or will study Law	No	We get 8 solutions if we remove this constraint.

In conclusion, the constraints that can be removed from the puzzle are:
1. One of them is going to London because it is an implicit constraint.
2. Oliver is not from South Africa as the model internally works such that South Africa cannot be Oliver’s nationality.

Task 2

Code File : da2.py

Note:

- The code is commented with sections Task A, B,.....H and the constraint that is being implemented.
- Explanations are also given for certain coding decisions.

Brief Outline of the Approach Used to solve the Problem:

1. Read the data from all the sheets in the excel file and store it as pandas dataframes.
2. The CP-SAT model is created.
3. The constraints are defined in a function called task_2_solution(). It returns the decision variables and a couple of other variables required to display the solutions.
4. The task_2_Solution does the following:
   - Create the projects_yn boolean decision variable for each project.
   - Read valid data(no blank spaces) from the ‘Projects’ sheet and store the data in the proj_mon_job dictionary.

- Read valid data from the ‘Quotes’ sheet and store the data in the cont_job_quote dictionary.
- Now, from the above dictionaries, create a new dictionary proj_mon_cont_job with valid combinations of the 4 variables – projects, months, contractors and jobs asthe key and a boolean decision variables as the values.
- All the constraints are mentioned in the assignment PDF are implemented with explanations in the code comments.
- An implicit constraint is implemented - If a project is taken, all jobs in the project should be completed. Explanation of this implementation is in the comments.

5. The solution printer is then called for solving the model.

Note:

- The code is generating too many solutions. So, the execution should be stopped with an interrupt.