Department of Computer Science
Technical University of Cluj-Napoca

# Artificial Intelligence

*Laboratory activity*

Name: Stoian Iulia Tia, Săsăran Andrei
Group: 30231

Email:

stoianiuliatia@gmail.com
sasaranandreipaul@gmail.com

Teaching Assistant:

Alexandru Ghiuruțan
alexandru.ghiurutan95@gmail.com

# Contents

Table 1: Lab scheduling

| **Activity** | **Deadline** |
|---|---|
| *Searching agents, Linux, Latex, Python, Pacman* | $W_1$ |
| *Uninformed search* | $W_2$ |
| *Informed Search* | $W_3$ |
| *Adversarial search* | $W_4$ |
| *Propositional logic* | $W_5$ |
| *First order logic* | $W_6$ |
| *Inference in first order logic* | $W_7$ |
| *Knowledge representation in first order logic* | $W_8$ |
| *Classical planning* | $W_9$ |
| *Contingent, conformant and probabilistic planning* | $W_{10}$ |
| *Multi-agent planing* | $W_{11}$ |
| *Modelling planning domains* | $W_{12}$ |
| *Planning with event calculus* | $W_{14}$ |

**Lab organisation.**

1. Laboratory work is 25% from the final grade.
2. There are three deliverables in total: 1. Search, 2. Logic, 3. Planning.
3. Before each deadline, you have to send your work (latex documentation/code) at moodle.cs.utcluj.ro
4. We use Linux and Latex
5. Plagiarism: Don't be a cheater! Cheating affects your colleagues, scholarships and a lot more.

# Chapter 1

# A1: Search

## 1.1 8 Puzzle - State Space Searches

Problema 8 Puzzle consta intr-un puzzle compus din (3 x 3 - 1) celule, numerotate de la 1 la 8. Cea de a 9-a celula care defineste puzzle-ul este o celula "goala" care este folosita de catre solver pentru a modifica compozitia puzzle-ului, prin mutarea unei celule adiacente in pozitia acesteia. Goal state-ul este atins atunci cand celula goala se afla pe prima pozitie (coltul din stanga sus), iar fiecare celula numerotata asezata in ordine crescatoare in continuarea acesteia, cu prioritate de la stanga la dreapta. Pentru a rezolva aceasta problema am utilizat doua tipuri de algoritmi de cautare neinformate (Breadth-First Search si Depth-First Search) si doua tipuri de algoritmi de cautare informate (Greedy Search si A* Search, fiecare utilizat pe rand cu una dintre euristicile Manhattan Distance si Euclidian Distance).

Fiecare forma in care pot fi asezate celulele puzzle-lului este un state space. Succesorii reprezinta state space-urile care rezulta prin realizarea unei mutari posibile (o mutare legala, care sa respecte cadranul puzzle-ului) a celulei goale, adica state space-ul curent are la randul lui ale state space-uri succesoare. In functie de algoritmul de cautare ales, aceste state space-uri vor fi parcurse pana la gasirea solutiei optime.

Proiectul este bazat pe modelul MVC (Model-View-Controller), avand o interfata prin intermediul careia utilizatorul poate avea acces la functionalitatile existente. Utilizatorul poate vedea efectiv celulele puzzle-ului, poate selecta sa se afiseze cum arata goal state-ul, poate apasa pe butonul "shuffle" care amesteca celulele intre ele si poate apasa pe butonul "solve" care va afisa in consola un string reprezentat prin concatenarea literelor N, E, S, W, reprezentand directiile in care a fost mutata celula goala.

# Chapter 2

# A2: Logics

## 2.1 Who owns the crocodile? - Easy Zebra Puzzle

Five girls are sitting in a row. Each girl has a favorite color, a chocolate bar, a pet, a hobby, and a place to go on holiday. Determine the name of the girl who owns the crocodile.

### 2.1.1 Clues

1. Jo likes the Wispa Bites.
2. The person with the Hamster likes Swimming.
3. Hannah eats Dairy Milk.
4. Jessica is on the left of Georgina.
5. Lucy is the first on the left.
6. The first person on the right likes Swimming.
7. The person who eats Milky Bars owns a Horse.
8. The person in the middle eats Dairy Milk.
9. Jessica likes Green.
10. The person on the left of the middle wants to go to Tobago.
11. The person who wants to go to Maldives likes Lilac.
12. The person who likes Wispa Bites sits next to the person who wants to go to Florida.
13. The person who likes Pink wants to go to Florida.
14. The person who sits first on the left likes lilac.
15. The girl that likes Blue owns a Puppy.
16. The person who likes Skiing sits next to the person who has a Hamster.
17. The girl on the right of the girl who likes Tennis likes Horse riding.
18. The girl next to the girl who likes Milky bars likes Boost.
19. The girl who likes Purple wants to go to Canada.
20. The girl who likes Crunchies owns a rabbit.
21. The girl who likes Skiing sits next to the girl who play Ten-Pin bowlings.
22. Jessica wants to go to Australia.

### 2.1.2 Human Thinking Solution

1. At the fifth position is the girl that likes swimming and has a hamster. **(Clue 6 and Clue 2)**
2. The girl at the third position is called Hannah and she eats Dairy Milk. **(Clue 8 and Clue 3)**
3. The girl at the first position is called Lucy. She likes lilac and wants to go to Maldives. **(Clue 5, Clue 14 and Clue 11)**
4. **Step 1** $\longrightarrow$ The fourth girl likes skiing and the third girl plays ten-pin bowling. **(Clue 16 and Clue 21)**
5. **Step 4** $\longrightarrow$ The first girl likes Tennis and the second girl likes horse riding. **(Clue 17)**
6. **Step 2 and Step 3** $\longrightarrow$ The fourth girl can only be Jessica and the fifth girl can only be Georgina. **(Clue 4)**
7. **Step 6** $\longrightarrow$ The second girl can only be Jo and she likes the Wispa Bites. **(Clue 1)**
8. **Step 7** $\longrightarrow$ The third girl wants to go to Florida and she likes pink. **(Clue 12 and Clue 13)**
9. The second girl wants to go to Tobago. **(Clue 10)**
10. **Step 6** $\longrightarrow$ The fourth girl wants to go to Australia and she likes green. **(Clue 22 and Clue 9)**
11. **Step 3, Step 7, Step 8 and Step 9** $\longrightarrow$ The fifth girl wants to go to Canada and she likes purple. **(Clue 19)**
12. **Step 3, Step 8, Step 10 and Step 11** $\longrightarrow$ The second girl likes blue and she owns a puppy. **(Clue 15)**
13. **Step 1, Step 2, Step 7 and Step 12** $\longrightarrow$ The fourth girl can only own a horse and she likes Milky Bars. The fifth girl likes Boost. **(Clue 18 and Clue 7)**
14. **Step 13** $\longrightarrow$ The first girl can only like Crunchies and she owns a rabbit. **(Clue 20)**
15. **Step 11** $\longrightarrow$ The third girl owns the crocodile.

### 2.1.3 Mathematical Solution

| | Girl #1 | Girl #2 | Girl #3 | Girl #4 | Girl #5 |
|---|---|---|---|---|---|
| Color | Lilac | Blue | Pink | Green | Purple |
| Name | Lucy | Jo | Hannah | Jessica | Georgina |
| Chocolate | Crunchies | Wispa Bites | Dairy Milk | Milky Bars | Boost |
| Pet | Rabbit | Puppy | Crocodile | Horse | Hamster |
| Hobby | Tennis | Horse Riding | Bowling | Skiing | Swimming |
| Holiday | Maldives | Tobago | Florida | Australia | Canada |

### 2.1.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Australia | 0 | 0 | 0 | 1 | 0 |
| Blue | 0 | 1 | 0 | 0 | 0 |
| Boost | 0 | 0 | 0 | 0 | 1 |
| Bowling | 0 | 0 | 1 | 0 | 0 |
| Canada | 0 | 0 | 0 | 0 | 1 |
| Crocodile | 0 | 0 | 1 | 0 | 0 |
| Crunchies | 1 | 0 | 0 | 0 | 0 |
| Dairy Milk | 0 | 0 | 1 | 0 | 0 |
| Florida | 0 | 0 | 1 | 0 | 0 |
| Georgina | 0 | 0 | 0 | 0 | 1 |
| Girl1 | 1 | 0 | 0 | 0 | 0 |
| Girl2 | 0 | 1 | 0 | 0 | 0 |
| Girl3 | 0 | 0 | 1 | 0 | 0 |
| Girl4 | 0 | 0 | 0 | 1 | 0 |
| Girl5 | 0 | 0 | 0 | 0 | 1 |
| Green | 0 | 0 | 0 | 1 | 0 |
| Hamster | 0 | 0 | 0 | 0 | 1 |
| Hannah | 0 | 0 | 1 | 0 | 0 |
| Horse | 0 | 0 | 0 | 1 | 0 |
| Jessica | 0 | 0 | 0 | 1 | 0 |
| Jo | 0 | 1 | 0 | 0 | 0 |
| Lilac | 1 | 0 | 0 | 0 | 0 |
| Lucy | 1 | 0 | 0 | 0 | 0 |
| Maldives | 1 | 0 | 0 | 0 | 0 |
| Milky Bars | 0 | 0 | 0 | 1 | 0 |
| Pink | 0 | 0 | 1 | 0 | 0 |
| Puppy | 0 | 1 | 0 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| Purple | 0 | 0 | 0 | 0 | 1 |
| Rabbit | 1 | 0 | 0 | 0 | 0 |
| Riding | 0 | 1 | 0 | 0 | 0 |
| Skiing | 0 | 0 | 0 | 1 | 0 |
| Swimming | 0 | 0 | 0 | 0 | 1 |
| Tennis | 1 | 0 | 0 | 0 | 0 |
| Tobago | 0 | 1 | 0 | 0 | 0 |
| Wispa Bites | 0 | 1 | 0 | 0 | 0 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

### 2.1.5 Prover9 Results

Proof from prover9, prooving line 2619 by using lines 502, 2615 and 2616.

```
502 -Florida(g3) | -Florida(g4).  [resolve(174,c,77,a)].
2615 Florida(g3).  [resolve(2605,b,2486,a),merge(b)].
2616 Florida(g4).  [back_unit_del(2609),unit_del(b,2615)].
2619 $F.  [back_unit_del(502),unit_del(a,2615),unit_del(b,2616)].
```

Back-unit deletion at line 502 $\Longrightarrow$ Copy clause 502.
**P or Q and —Q $\longrightarrow$ P**

Q = —Florida(g4)
—Q = Florida(g4)
$\longrightarrow$ **P = —Florida(g3)**

Unit-deletion at line 2615 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 2615 (which is a unit clause).

Unit-deletion at line 2616 $\Longrightarrow$ The second literal **b** has been removed because it was an instance of the negation clause 2616 (which is a unit clause).

**Q and —Q or R$\longrightarrow$ R**

Q = Florida(g3)
—Q = —Florida(g3)
$\longrightarrow$ **R = Florida(g4)**

## 2.2 Ancient Civilizations - Medium Zebra Puzzle

Five girls are side by side talking about ancient civilizations. Each girl is passionate about a certain ancient civilization and knows a lot of things regarding a specific subject of it. Follow the clues to figure out which girl likes the Aztecs.

### 2.2.1 Clues

1. At the fourth position is the girl drinking Strawberry juice.
2. The girl drinking Strawberry juice is somewhere between the girl who likes Art and the girl that is drinking Apple juice, in that order.
3. Riley is next to the girl who likes the Roman civilization.
4. The girl wearing the Green shirt is drinking Apple juice.
5. At the fourth position is the girl that is interested in Religion.
6. The 12 years old girl is exactly to the right of the 11 years old girl.
7. At one of the ends is the girl who likes the Egyptian civilization.
8. The girl wearing the Black shirt is somewhere between the girl that likes Science and the girl wearing the Red shirt, in that order.
9. The girl wearing the Red shirt is somewhere to the left of the girl wearing the White shirt.
10. The girl drinking Cranberry juice likes the Greek civilization.
11. Ella is at the third position.
12. The 12 years old girl is next to the girl that is interested in Architecture.
13. Riley is exactly to the right of the girl that admires the Roman civilization.
14. The girl drinking Cranberry juice is somewhere between the 13 years old girl and the girl drinking Grapefruit juice, in that order.
15. At one of the ends is the girl that likes Science.
16. At one of the ends is the girl who loves Architecture.
17. Lily is exactly to the right of the girl who likes Economy.
18. Makayla is 10.
19. The girl interested in the Aztec civilization is somewhere between the girl interested in the Roman civilization and the girl interested in the Chinese civilization, in that order.
20. The youngest girl is at one of the ends.
21. At the second position is the girl that likes Art.

### 2.2.2 Human Thinking Solution

1. The second girl likes art, the fourth girl drinks strawberry juice, while the fifth girl drinks apple juice and has a green shirt. **(Clue 1, Clue 2, Clue 4 and Clue 21)**
2. The first girl likes science and the fifth girl likes architecture. **(Clue 8, Clue 15 and Clue 16)**
3. **Step 1 and Step 2** $\longrightarrow$ The second girl is wearing a black shirt, the third girl is wearing a red shirt and the fourth girl is wearing a white shirt. **(Clue 8)**
4. **Step 3** $\longrightarrow$ The first girl is wearing a blue shirt.
5. **Step 1 and Step 2** $\longrightarrow$ The third girl is interested in Economy and the fourth girl is interested in religion. **(Clue 5)**
6. **Step 5** $\longrightarrow$ The fourth girl is called Lily. **(Clue 17)**
7. **Step 6** $\longrightarrow$ The first girl likes the Roman civilization, the second girl is called Riley and the third girl is called Ella. **(Clue 11 and Clue 13)**
8. **Step 7** $\longrightarrow$ The fifth girl likes the Egyptian civilization. **(Clue 7)**
9. **Step 2** $\longrightarrow$ The fourth girl is 12 years old. **(Clue 12)**
10. **Step 9** $\longrightarrow$ The third girl is 11 years old. **(Clue 6)**
11. **Step 1** $\longrightarrow$ The first girl is 13 years old, the second girl drinks cranberry juice and the third girl drinks grapefruit juice. **(Clue 14)**
12. **Step 11** $\longrightarrow$ The first girl can only drink orange juice.
13. **Step 11** $\longrightarrow$ The fifth girl is called Makayla and she is 10 years old. **(Clue 18 and Clue 20)**
14. **Step 13** $\longrightarrow$ The first girl can only be named Julia and the second girl can only be 14 years old.
15. **Step 11** $\longrightarrow$ The second girl likes the Greek civilization. **(Clue 10)**
16. **Step 7, Step 8, Step 15** $\longrightarrow$ The third girl likes the Aztec civilization and the fourth girl likes the Chinese civilization. **(Clue 19)**

### 2.2.3 Mathematical Solution

| | Girl #1 | Girl #2 | Girl #3 | Girl #4 | Girl #5 |
|---|---|---|---|---|---|
| Shirt | Blue | Black | Red | White | Green |
| Name | Julia | Riley | Ella | Lily | Makayla |
| Civilization | Roman | Greek | Aztec | Chinese | Egyptian |
| Subject | Science | Art | Economy | Religion | Architecture |
| Age | 13 years | 14 years | 11 years | 12 years | 10 years old |
| Juice | Orange | Cranberry | Grapefruit | Strawberry | Apple |

### 2.2.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Apple | 0 | 0 | 0 | 0 | 1 |
| Architecture | 0 | 0 | 0 | 0 | 1 |
| Art | 0 | 1 | 0 | 0 | 0 |
| Aztec | 0 | 0 | 1 | 0 | 0 |
| Black | 0 | 1 | 0 | 0 | 0 |
| Blue | 1 | 0 | 0 | 0 | 0 |
| Chinese | 0 | 0 | 0 | 1 | 0 |
| Cranberry | 0 | 1 | 0 | 0 | 0 |
| Economy | 0 | 0 | 1 | 0 | 0 |
| Egyptian | 0 | 0 | 0 | 1 | 0 |
| Eleven | 0 | 0 | 1 | 0 | 0 |
| Ella | 0 | 0 | 1 | 0 | 0 |
| Fourteen | 0 | 1 | 0 | 0 | 0 |
| Girl1 | 1 | 0 | 0 | 0 | 0 |
| Girl2 | 0 | 1 | 0 | 0 | 0 |
| Girl3 | 0 | 0 | 1 | 0 | 0 |
| Girl4 | 0 | 0 | 0 | 1 | 0 |
| Girl5 | 0 | 0 | 0 | 0 | 1 |
| Grapefruit | 0 | 0 | 1 | 0 | 0 |
| Greek | 0 | 1 | 0 | 0 | 0 |
| Green | 0 | 0 | 0 | 0 | 1 |
| Julia | 1 | 0 | 0 | 0 | 0 |
| Lily | 0 | 0 | 0 | 1 | 0 |
| Makayla | 0 | 0 | 0 | 0 | 1 |
| Orange | 1 | 0 | 0 | 0 | 0 |
| Red | 0 | 0 | 0 | 1 | 0 |
| Religion | 0 | 0 | 1 | 0 | 0 |
| Riley | 0 | 1 | 0 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| Roman | 0 | 0 | 0 | 0 | 1 |
| Science | 1 | 0 | 0 | 0 | 0 |
| Strawberry | 0 | 1 | 0 | 0 | 0 |
| Ten | 0 | 0 | 0 | 1 | 0 |
| Thirteen | 0 | 0 | 0 | 0 | 1 |
| Twelve | 1 | 0 | 0 | 0 | 0 |
| White | 0 | 1 | 0 | 0 | 0 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| inbetween | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.2.5 Prover9 Results

Proof from prover9, prooving line 2776 by using lines 2775 and 160.

```
160 -inbetween(g2,g2).  [assumption].
2775 inbetween(g2,x).  [back_unit_del(582),unit_del(a,2774)].
2776 $F.  [resolve(2775,a,160,a)].
```

Resolve at line 2776 $\Longrightarrow$ resolve the first literal of clause 2775 with the first literal of clause 160.

## 2.3 Persian Rugs - Hard Zebra Puzzle

Five men, who just bought Persian rugs, are side by side. Each one bought a very specific Persian rug for his home. Try to find out which are the characteristics from Timothy's new rug.

### 2.3.1 Clues

1. The man who bought the 400 dollars rug is exactly to the left of the man who bought the rug with Birds.
2. Bryan is next to the client that got the Runner rug.
3. The Red rug's owner is somewhere to the left of the man who bought a rug for his Hall.
4. The buyer who got the most expensive rug is next to the buyer who got the Round rug.
5. The Yellow rug's owner is somewhere between the Oval rug's owner and the man that spent 800 dollars, in that order.
6. The man that bought the Red rug is exactly to the left of the man that bought a rug for his Office.
7. Garrett is next to the customer who spent 600 dollars.
8. Shane is somewhere to the right of the Orange rug's owner.
9. The cheapest rug has Stars on it.
10. At the fifth position is the client who bought the Rectangular rug.
11. The man who bought the 600 dollars rug is exactly to the left of the man who bought the rug with Roses.
12. Casey is somewhere to the right of the customer that got the Yellow rug.
13. The buyer who purchased the Round rug is exactly to the left of the man who spent 800 dollars.
14. The man that bought the cheapest rug is exactly to the left of the man that acquired the Square rug.
15. At the third position is the man that got a rug for his Bedroom.
16. The customer that bought the rug with Stars is exactly to the right of the customer that bought the Green rug.
17. Shane is next to the man who acquired the Round rug.
18. The man who got the rug with Birds is somewhere between the man who got the Yellow rug and the man who got the rug with Lotuses, in that order.
19. The customer that spent 800 dollars is somewhere to the right of the Red rug's owner.
20. The client that bought a rug for his Living room is exactly to the right of the client that bought a rug for his Library.

### 2.3.2 Human Thinking Solution

1. The man that bought the green rug, the man that bought the 400 dollars rug that has stars, and the man that bought the square rug that has birds are next to each other, in this order. **(Clue 9, Clue 10 and Clue 11)**
2. The man that bought the rug for his office can only be directly to the left of the man that bought the rug for his hall. **(Clue 15, Clue 20, Clue 6 and Clue 3)**
3. **Step 2** $\longrightarrow$ The red rug is the third, the man that bought the rug for his office is the fourth and the man that bought the rug for his hall is the fifth. **(Clue 6)**
4. **Step 3** $\longrightarrow$ The that bought the rug for his living room is the first and the man that bought the rug for his library is the second. **(Clue 20 and Clue 15)**
5. The man that spent 1.200 dollars on the rug is directly to the left of the man that bought the round rug. **(Clue 13 and Clue 4)**
6. **Step 2 and Step 5** $\longrightarrow$ The third man bought the 1.200 dollars red rug, the fourth man bought the round rug and the fifth man bought the 800 dollars rug. **(Clue 19)**
7. **Step 1 and Step 6** $\longrightarrow$ The first man bought the green rug, the second man bought the 400 dollars rug that has stars and the third man bought the square rug that has birds.
8. **Step 7** $\longrightarrow$ The second man bought the yellow rug. **(Clue 18)**
9. **Step 8** $\longrightarrow$ The first man bought the oval rug. **(Clue 5)**
10. **Step 9** $\longrightarrow$ The second man bought the runner rug.
11. **Step 6 and Step 7** $\longrightarrow$ The fourth man spent 600 dollars on the rug and the fifth man bought the rug that has roses. **(Clue 11)**
12. **Step 11** $\longrightarrow$ the first man bought the 1.000 dollars rug.
13. **Step 7 and Step 11** $\longrightarrow$ The fourth man bought the rug that has lotuses. **(Clue 18)**
14. **Step 13** $\longrightarrow$ The first man bought the rug that has diamonds.
15. **Step 7, Step 8 and Step 3** $\longrightarrow$ The fourth man bought the orange rug and the fifth man is called Shane. **(Clue 8)**
16. **Step 15** $\longrightarrow$ The fifth man bought the blue rug.
17. **Step 15** $\longrightarrow$ The third man is called Garrett. **(Clue 7)**
18. **Step 15 and Step 17** $\longrightarrow$ The fourth man is called Casey. **(Clue 12)**
19. **Step 15 and Step 10** $\longrightarrow$ The first man is called Bryan. **(Clue 2)**
20. **Step 19** $\longrightarrow$ The second man is called Timothy.

### 2.3.3 Mathematical Solution

### 2.3.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Bedroom | 0 | 0 | 1 | 0 | 0 |
| Birds | 0 | 0 | 1 | 0 | 0 |
| Blue | 0 | 0 | 0 | 0 | 1 |
| Bryan | 1 | 0 | 0 | 0 | 0 |
| Casey | 0 | 0 | 0 | 1 | 0 |
| Diamonds | 1 | 0 | 0 | 0 | 0 |
| EightHundred | 0 | 0 | 0 | 0 | 1 |
| FourHundred | 0 | 1 | 0 | 0 | 0 |
| Garrett | 0 | 0 | 1 | 0 | 0 |
| Green | 1 | 0 | 0 | 0 | 0 |
| Hall | 0 | 0 | 0 | 0 | 1 |
| Library | 1 | 0 | 0 | 0 | 0 |
| LivingRoom | 0 | 1 | 0 | 0 | 0 |
| Lotuses | 0 | 0 | 0 | 1 | 0 |
| Man1 | 1 | 0 | 0 | 0 | 0 |
| Man2 | 0 | 1 | 0 | 0 | 0 |
| Man3 | 0 | 0 | 1 | 0 | 0 |
| Man4 | 0 | 0 | 0 | 1 | 0 |
| Man5 | 0 | 0 | 0 | 0 | 1 |
| Office | 0 | 0 | 0 | 1 | 0 |
| OneThousan | 1 | 0 | 0 | 0 | 0 |
| Orange | 0 | 0 | 0 | 1 | 0 |
| Oval | 1 | 0 | 0 | 0 | 0 |
| Rectangular | 0 | 0 | 0 | 0 | 1 |
| Red | 0 | 0 | 1 | 0 | 0 |
| Roses | 0 | 0 | 0 | 0 | 1 |
| Round | 0 | 0 | 0 | 1 | 0 |
| Runner | 0 | 1 | 0 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| Shane | 0 | 0 | 0 | 0 | 1 |
| SixHundred | 0 | 0 | 0 | 1 | 0 |
| Square | 0 | 0 | 1 | 0 | 0 |
| Stars | 0 | 1 | 0 | 0 | 0 |
| Timothy | 0 | 1 | 0 | 0 | 0 |
| TwelveHundred | 0 | 0 | 1 | 0 | 0 |
| Yellow | 0 | 1 | 0 | 0 | 0 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.3.5 Prover9 Results

Proof from prover9, prooving line 11474 by using lines 11471, 10628 and 126.

```
126 -right(d2,d1).  [assumption].
10628 EightHundred(d1).  [resolve(10619,b,10600,b),merge(c),unit_del(b,131)].
11471 -EightHundred(x) | right(d2,x).  [resolve(11364,a,191,b)].
11474 $F.  [resolve(11471,a,10628,a),unit_del(a,126)].
```

Resolve at line 11474 $\Longrightarrow$ resolve the first literal of clause 11471 with the first literal of clause 10628.

Unit-deletion at line 126 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 126 (which is a unit clause).

## 2.4 Travel Agency - Hard Zebra Puzzle

Five women are side by side booking a trip in a travel agency. Each one is traveling to a different country. Follow the clues to discover where each one is going.

### 2.4.1 Clues

1. The Singer is at the third position.
2. The woman traveling for 20 days is somewhere between the woman who is going to Peru and the owner of the Blue purse, in that order.
3. Ana is exactly to the left of the Biologist.
4. The 32 years old is going to see the Sahara.
5. The owner of the White purse is exactly to the right of the woman traveling to visit Machu Picchu.
6. Glenda is somewhere to the right of the woman who has the Green purse.
7. The person wearing the White purse is somewhere between the 30 years old woman and the owner of the Blue purse, in that order.
8. The 24 years old woman is going to visit an Aztec pyramid.
9. The woman wearing the White purse is somewhere to the left of the youngest woman.
10. The traveler going to Italy is exactly to the right of the woman traveling for 20 days.
11. The person who is going to travel for 25 days has the Red purse.
12. The Judge is in the first position.
13. The Nurse is exactly to the right of the woman who is going to travel for 20 days.
14. The Hostess is somewhere between Lara and the woman who has the Blue purse, in that order.
15. In the second position is the woman that is going to travel for 15 days.
16. Rose has the Green purse.
17. The woman who is traveling for less than a week is exactly to the left of the 32 years old woman.
18. The person traveling for 5 days is 28.
19. The Blue purse owner is somewhere between the 30 years old woman and the owner of the Yellow purse, in that order.

### 2.4.2 Human Thinking Solution

1. The third woman is traveling for 20 days, the fourth woman has the blue purse and the fifth woman has the yellow purse. **(Clue 15, Clue 2 and Clue 19)**
2. The first woman is called Lara, the second woman is a hostess. **(Clue 12, Clue 1 and Clue 14)**
3. The fourth woman is a nurse. **(Clue 13 and Clue 1)**
4. **Step 3** $\longrightarrow$ The fifth woman can only be a biologist.
5. **Step 1 ans Step 2** $\longrightarrow$ The first woman is traveling for 25 days and has a red purse. **(Clue 11, Clue 1 and Clue 15)**
6. **Step 1 and Step 5** $\longrightarrow$ The fourth woman is 28 years old and is traveling for 5 days, while the fifth woman is 32 years old and is traveling to Egypt. **(Clue 17, Clue 18, Clue 4 and Clue 15)**
7. **Step 6** $\longrightarrow$ The fifth woman can only be travelling for 10 days.
8. **Step 4** $\longrightarrow$ The fourth woman is called Ana. **(Clue 3)**
9. **Step 1 and Step 6** $\longrightarrow$ The first woman is 30 years old, the second woman has the white purse and the third woman is 24 years old. **(Clue 7 and Clue 9)**
10. **Step 9** $\longrightarrow$ The third woman can only have the green purse.
11. **Step 10** $\longrightarrow$ The third woman is called Rose. **(Clue 14)**
12. **Step 9** $\longrightarrow$ The second woman can only be 26 years old.
13. **Step 9** $\longrightarrow$ The first woman is going to Peru. **(Clue 5)**
14. **Step 9** $\longrightarrow$ The third woman is going to Mexico. **(Clue 8)**
15. **Step 1** $\longrightarrow$ The fourth woman is travelling to Italy. **(Clue 10)**
16. **Step 15** $\longrightarrow$ The second woman can only be travelling to China.
17. **Step 10 and Step 8** $\longrightarrow$ The fifth woman is called Glenda. **(Clue 6)**
18. **Step 17** $\longrightarrow$ The second woman can only be called Jessie.

### 2.4.3 Mathematical Solution

| | Woman #1 | Woman #2 | Woman #3 | Woman #4 | Woman #5 |
|---|---|---|---|---|---|
| Purse | Red | White | Green | Blue | Yellow |
| Name | Lara | Jessie | Rose | Ana | Glenda |
| Age | 30 years | 26 years | 24 years | 28 years | 32 years |
| Profession | Judge | Hostess | Singer | Nurse | Biologist |
| Country | Peru | China | Mexico | Italy | Egypt |
| Duration | 25 days | 15 days | 20 days | 5 days | 10 days |

### 2.4.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Ana | 0 | 0 | 0 | 1 | 0 |
| Biologist | 0 | 0 | 0 | 0 | 1 |
| Blue | 0 | 0 | 0 | 1 | 0 |
| China | 0 | 1 | 0 | 0 | 0 |
| Egypt | 0 | 0 | 0 | 0 | 1 |
| FifteenDays | 0 | 1 | 0 | 0 | 0 |
| FiveDays | 0 | 0 | 0 | 1 | 0 |
| Glenda | 0 | 0 | 0 | 0 | 1 |
| Green | 0 | 0 | 1 | 0 | 0 |
| Hostess | 0 | 1 | 0 | 0 | 0 |
| Italy | 0 | 0 | 0 | 1 | 0 |
| Jessie | 0 | 1 | 0 | 0 | 0 |
| Judge | 1 | 0 | 0 | 0 | 0 |
| Lara | 1 | 0 | 0 | 0 | 0 |
| Mexico | 0 | 0 | 1 | 0 | 0 |
| Nurse | 0 | 0 | 0 | 1 | 0 |
| Peru | 1 | 0 | 0 | 0 | 0 |
| Red | 1 | 0 | 0 | 0 | 0 |
| Rose | 0 | 0 | 1 | 0 | 0 |
| Singer | 0 | 0 | 1 | 0 | 0 |
| TenDays | 0 | 0 | 0 | 0 | 1 |
| Thirty | 1 | 0 | 0 | 0 | 0 |
| ThirtyTwo | 0 | 0 | 0 | 0 | 1 |
| TwentyDays | 0 | 0 | 1 | 0 | 0 |
| TwentyEight | 0 | 0 | 0 | 1 | 0 |
| TwentyFiveDays | 1 | 0 | 0 | 0 | 0 |
| TwentyFour | 0 | 0 | 1 | 0 | 0 |
| TwentySix | 0 | 1 | 0 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| White | 0 | 1 | 0 | 0 | 0 |
| Woman1 | 1 | 0 | 0 | 0 | 0 |
| Woman2 | 0 | 1 | 0 | 0 | 0 |
| Woman3 | 0 | 0 | 1 | 0 | 0 |
| Woman4 | 0 | 0 | 0 | 1 | 0 |
| Woman5 | 0 | 0 | 0 | 0 | 1 |
| Yellow | 0 | 0 | 0 | 0 | 1 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.4.5 Prover9 Results

Proof from prover9, prooving line 14449 by using lines 14448, 12412 and 130.

```
130 -right(d1,d1).  [assumption].
12412 Blue(d1).  [resolve(12411,b,12044,b),merge(b)].
14448 -Blue(x) | right(d1,x).  [resolve(14338,a,13504,a)].
14449 $F.  [resolve(14448,a,12412,a),unit_del(a,130)].
```

Resolve at line 14448 $\Longrightarrow$ resolve the first literal of clause 14448 with the first literal of clause 12412.

Unit-deletion at line 130 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 130 (which is a unit clause).

## 2.5 Secret Agents - Very Hard Zebra Puzzle

Five spies are side by side in a briefing room getting instructions for their next missions. Each spy has an exclusive accessory, a special skill and is going to travel to a determined country. Which one is a martial arts expert?

### 2.5.1 Clues

1. Austin is next to the agent wearing the Black tie.
2. The master of Disguise is exactly to the right of the agent that has a spy Umbrella.
3. The 35 years old agent is going to a mission on Tripoli.
4. James is the youngest secret agent.
5. The agent that is going to Australia is next to the agent that is specialized in Parkour.
6. James is exactly to the right of the agent that has a special Clock.
7. The spy that has an unique Umbrella is between the 40 years old agent and Austin, in that order.
8. Stan is next to the agent that is going to Asia.
9. Sterling is at one of the ends.
10. The man wearing the Red tie is 40 years old.
11. The spy that is going to South America is exactly to the left of the 45 years old spy.
12. Jason is exactly to the left of Austin.
13. The Driver expert is next to the 30 years old man.
14. The 35 years old agent is next to the agent that is going to Sydney.
15. The agent with advanced knowledge of Hacking is exactly to the left of the 35 years old man.
16. The spy wearing the Purple tie is next to the Geek spy.
17. Austin is 30.
18. The agent that has a special Phone is exactly to the left of the agent that is going to Africa.
19. The agent wearing a spy Ring is somewhere to the right of the agent wearing the Purple tie.
20. At the second position is the spy wearing the Green tie.
21. The spy that is going to Australia is exactly to the right of the 30 years old spy.

### 2.5.2 Human Thinking Solution

1. The first agent is 40 years old, the second agent is called Jason and has a unique umbrella, the third agent is called Austin and he is 30 years old, the fourth agent has advanced knowledge in computer hacking and is going to Australia, the fifth agent is 35 years old and is going to Libya. **(Clue 12, Clue 17, Clue 21, Clue 3, Clue 14, Clue 15 and Clue 7)**
2. **Step 1** $\longrightarrow$ The fourth agent is wearing the black tie. **(Clue 20 and Clue 1)**
3. **Step 1** $\longrightarrow$ The third agent is a master of disguise. **(Clue 2)**
4. **Step 1** $\longrightarrow$ The fourth agent is called James and he is 25 years old. **(Clue 4)**
5. **Step 4** $\longrightarrow$ The second agent can only be 45 years old.
6. **Step 1 and Step 3** $\longrightarrow$ The fifth agent is specialized in parkour. **(Clue 5)**
7. **Step 4** $\longrightarrow$ The third agent has a special clock. **(Clue 6)**
8. **Step 1 and Step 4** $\longrightarrow$ The first agent is called Stan and the second agent is going to Russia. **(Clue 8)**
9. **Step 8** $\longrightarrow$ The fifth agent is called Sterling. **(Clue 9)**
10. **Step 1** $\longrightarrow$ The first agent is wearing the red tie. **(Clue 10)**
11. **Step 5** $\longrightarrow$ The first agent is going to Brazil. **(Clue 11)**
12. **Step 1** $\longrightarrow$ The second agent is an expert driver. **(Clue 13)**
13. **Step 12** $\longrightarrow$ The first agent can only be specialized in martial arts.
14. **Step 11** $\longrightarrow$ The third agent can only be going to Germany.
15. **Step 1** $\longrightarrow$ The fourth agent has a special phone. **(Clue 18)**
16. **Step 1, Step 7, Step 15, Step 10, Step 20 and Step 2** $\longrightarrow$ The third agent is wearing the purple tie and the fifth agent is wearing the spy ring. **(Clue 19)**
17. **Step 16** $\longrightarrow$ The first agent can only have a special pen and the fifth agent can only be wearing the blue tie.

### 2.5.3 Mathematical Solution

| | Agent #1 | Agent #2 | Agent #3 | Agent #4 | Agent #5 |
|---|---|---|---|---|---|
| Tie | Red | Green | Purple | Black | Blue |
| Name | Stan | Jason | Austin | James | Sterling |
| Country | Brazil | Russia | Germany | Australia | Libya |
| Accessory | Pen | Umbrella | Clock | Phone | Ring |
| Skill | Martial Arts | Driving | Disguise | Computer Hacking | Parkour |
| Age | 40 years | 45 years | 30 years | 25 years | 35 years |

### 2.5.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Agent1 | 1 | 0 | 0 | 0 | 0 |
| Agent2 | 0 | 1 | 0 | 0 | 0 |
| Agent3 | 0 | 0 | 1 | 0 | 0 |
| Agent4 | 0 | 0 | 0 | 1 | 0 |
| Agent5 | 0 | 0 | 0 | 0 | 1 |
| Austin | 0 | 0 | 1 | 0 | 0 |
| Australia | 0 | 0 | 0 | 1 | 0 |
| Black | 0 | 0 | 0 | 1 | 0 |
| Blue | 0 | 0 | 0 | 0 | 1 |
| Brazil | 1 | 0 | 0 | 0 | 0 |
| Clock | 0 | 0 | 1 | 0 | 0 |
| ComputerHacking | 0 | 0 | 0 | 1 | 0 |
| Disguise | 0 | 0 | 1 | 0 | 0 |
| Driving | 0 | 1 | 0 | 0 | 0 |
| Forty | 1 | 0 | 0 | 0 | 0 |
| FortyFive | 0 | 1 | 0 | 0 | 0 |
| Germany | 0 | 0 | 1 | 0 | 0 |
| Green | 0 | 1 | 0 | 0 | 0 |
| James | 0 | 0 | 0 | 1 | 0 |
| Jason | 0 | 1 | 0 | 0 | 0 |
| Libya | 0 | 0 | 0 | 0 | 1 |
| MartialArts | 1 | 0 | 0 | 0 | 0 |
| Parkour | 0 | 0 | 0 | 0 | 1 |
| Pen | 1 | 0 | 0 | 0 | 0 |
| Phone | 0 | 0 | 0 | 1 | 0 |
| Purple | 0 | 0 | 1 | 0 | 0 |
| Red | 1 | 0 | 0 | 0 | 0 |
| Ring | 0 | 0 | 0 | 0 | 1 |

| Russia | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|
| Stan | 1 | 0 | 0 | 0 | 0 |
| Sterling | 0 | 0 | 0 | 0 | 1 |
| Thirty | 0 | 0 | 1 | 0 | 0 |
| ThirtyFive | 0 | 0 | 0 | 0 | 1 |
| TwentyFive | 0 | 0 | 0 | 1 | 0 |
| Umbrella | 0 | 1 | 0 | 0 | 0 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.5.5 Prover9 Results

Proof from prover9, prooving line 24218 by using lines 24217, 18365 and 99.

```
99 -directlyright(d3,d1).  [assumption].
18365 Austin(d1).  [resolve(18364,b,17649,b),merge(b)].
24217 -Austin(x) | directlyright(d3,x).  [resolve(24176,a,187,a)].
24218 $F.  [resolve(24217,a,18365,a),unit_del(a,99)].
```

Resolve at line 24218 $\Longrightarrow$ resolve the first literal of clause 24217 with the first literal of clause 18365.

Unit-deletion at line 99 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 99 (which is a unit clause).

## 2.6 Who likes dolphins? - Easy Zebra Puzzle

There are four houses in a line. Each house has a number, a color and an owner. Each owner has a nationality, a favourite animal and sport. Find out whose favourite animals are dolphins.

### 2.6.1 Clues

1. The second house is Black.
2. The British lives in the first house.
3. There are two houses between the person who likes Bowling and the person who likes Swimming.
4. The American live directly to the left of the person who likes Turtles.
5. There are two houses between the person who likes Horses and the person who likes Butterflies on the right.
6. The person who likes Bowling lives somewhere to the right of the person who likes Tennis.
7. There is one house between the Irish and the person who likes Handball on the left.
8. There is one house between the person who likes Horses and the Red house on the right.
9. There is one house between the person who likes Handball and the White house on the right.

### 2.6.2 Human Thinking Solution

1. The person who likes Horses lives in the first house and the person who likes Butterflies lives in the last house. **(Clue 5)**
2. The person who likes Tennis lives in the second or third house and the person who likes Bowling lives in the first or last house. The person who likes Handball lives in the second or third house. **(Clue 3)**
3. **Step 2** $\longrightarrow$ The person who likes Bowling lives in the last house and the person who likes Swimming lives in the first house. **(Clue 6 and Clue 3)**
4. **Step 2** $\longrightarrow$ The person who likes Handball lives in the second house and the person who lives in the last house is Irish. The American person lives in the second or third house. **(Clue 7)**
5. **Step 4** $\longrightarrow$ The American Person lives in the second house and the person who likes Turtles lives in the third house. **(Clue 4)**
6. **Step 5** $\longrightarrow$ The person who likes dolphins lives in the second house.

### 2.6.3 Mathematical Solution

| | House #1 | House #2 | House #3 | House #4 |
|---|---|---|---|---|
| Color | Blue | Black | Red | White |
| Nationality | British | American | Canadian | Irish |
| Animal | Horses | Dolphins | Turtles | Butterflies |
| Sport | Swimming | Handball | Tennis | Bowling |

### 2.6.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | |
|---|---|---|---|---|
| American | 0 | 1 | 0 | 0 |
| Black | 0 | 1 | 0 | 0 |
| Blue | 1 | 0 | 0 | 0 |
| Bowling | 1 | 0 | 0 | 0 |
| British | 1 | 0 | 0 | 0 |
| Butterflies | 0 | 0 | 0 | 1 |
| Canadian | 0 | 0 | 1 | 0 |
| Dolphins | 0 | 1 | 0 | 0 |
| Handball | 0 | 1 | 0 | 0 |
| Horses | 1 | 0 | 0 | 0 |
| HouseFour | 0 | 0 | 0 | 1 |
| HouseOne | 1 | 0 | 0 | 0 |
| HouseThree | 0 | 0 | 1 | 0 |
| HouseTwo | 0 | 1 | 0 | 0 |
| Irish | 0 | 0 | 0 | 1 |
| Red | 0 | 1 | 0 | 0 |
| Swimming | 0 | 0 | 1 | 0 |
| Tennis | 0 | 0 | 1 | 0 |
| Turtles | 0 | 0 | 1 | 0 |
| White | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 |
| 3 | 1 | 1 | 1 | 0 |

| inbetween | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |

| middleneighbour | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |

| right | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 |

| rightneighbour | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 |

### 2.6.5 Prover9 Results

Proof from prover9, prooving line 514 by using lines 513, 490 and 107.

```
107 -inbetween(h4,h2).  [assumption].
490 Bowling(h4).  [resolve(470,b,476,b),merge(c),unit_del(b,106)].
513 -Bowling(x) | inbetween(x,h2).  [resolve(506,a,137,b)].
514 $F.  [resolve(513,a,490,a),unit_del(a,107)].
```

Resolve at line 514 $\Longrightarrow$ resolve the first literal of clause 513 with the first literal of clause 490.

Unit-deletion at line 107 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 107 (which is a unit clause).

## 2.7 Hardware Store - Medium Zebra Puzzle

Five men are next to each other at a hardware store. Each one is buying a specific tool and getting a discount. Solve this logic problem to discover these information.

### 2.7.1 Clues

1. The man wearing the Yellow shirt is somewhere to the left of the Mechanic.
2. Billy is buying a tool with 20 percent off.
3. The guy buying a tool with 15 percent discount is at one of the ends.
4. The 45 years old man is somewhere between the Plumber and the 35 years old man, in that order.
5. The one buying a tool with the biggest discount is next to the 50 years old man.
6. Dennis is wearing the Green shirt.
7. The 50 years old man is in the middle.
8. Larry is next to the man that is buying a tool with the smallest discount.
9. The one buying a Hammer is somewhere between the man buying the Wrench and the man buying the Screwdriver, in that order.
10. The Technician is at the third position.
11. The Plumber is next to the man wearing the Green shirt.
12. The oldest man is buying a Drill.
13. The guy wearing the White shirt is somewhere to the right of the man wearing the Blue shirt.
14. The man buying a Hammer is somewhere between the one who is getting 15 percent discount and the one buying a Saw, in that order.
15. George is next to the Electrician.
16. The man buying tools with 5 percent and 25 percent off are next to each other.
17. Philip is buying a Saw.
18. At the second position is the man buying a Drill.
19. The man wearing the Yellow shirt is next to the one buying a Saw.
20. The guy getting the best discount is somewhere between the 55 years old man and the man buying a tool with 10 percent off, in that order.
21. The man wearing the Red shirt is next to the man buying a tool with 15

### 2.7.2 Human Thinking Solution

1. The second man is 55 years old. **(Clue 18 and Clue 12)**
2. **Step 1** $\longrightarrow$ The fourth man is 45 years old and the fifth person is 35 years old. **(Clue 4 and Clue 7)** 3. **Step 2** $\longrightarrow$ The first man is 40 years old.
4. The first man has a 15 percent discount. **(Clue 3 and Clue 14)**
5. **Step 4** $\longrightarrow$ The second man is wearing the red shirt. **(Clue 21)**
6. **Step 2** $\longrightarrow$ The first man is called Dennis and he is wearing the green shirt, while the second man is a plumber. **(Clue 10, Clue 6 and Clue 4)**
7. **Step 10 and Step 6** $\longrightarrow$ The fourth man is wearing the yellow shirt and the fifth man is a mechanic. **(Clue 1)**
8. **Step 5, Step 6 and Step 7** $\longrightarrow$ The third man is wearing the blue shirt and the fifth man is wearing the white shirt. **(Clue 13)**
9. **Step 7 and Step 4** $\longrightarrow$ The fifth man is called Philip and he is buying a saw. **(Clue 19, Clue 17 and Clue 14)**
10. The fourth man has a 25 percent discount. **(Clue 7, Clue 16 and Clue 20)**
11. **Step 1 and Step 10** $\longrightarrow$ The fifth man has a 10 percent discount. **(Clue 20**
12. **Step 10 and Step 11** $\longrightarrow$ The third man has a 5 percent discount. **(Clue 16)**
13. **Step 12** $\longrightarrow$ The second man can only have a 20 percent discount.
14. **Step 13** $\longrightarrow$ The second man is called Billy. **Clue 2**
15. **Step 14 and Step 2** $\longrightarrow$ The fourth man is called Larry. **(Clue 8)**
16. **Step 15** $\longrightarrow$ The third man can only be called George.
17. **Step 16 and Step 6** $\longrightarrow$ The fourth man is an electrician. **(Clue 15)**
18. **Step 17** $\longrightarrow$ The first man can only be an engineer.
19. **Step 9** $\longrightarrow$ The first man is buying a wrench, the second man is buying a hammer and the fourth man is buying a screwdriver. **(Clue 9, Clue 14 and Clue 18**

### 2.7.3 Mathematical Solution

| | Man #1 | Man #2 | Man #3 | Man #4 | Man #5 |
|---|---|---|---|---|---|
| Shirt | Green | Red | Blue | Yellow | White |
| Name | Dennis | Billy | George | Larry | Philip |
| Tool | Wrench | Drill | Hammer | Screwdriver | Saw |
| Discount | 15% | 20% | 5% | 25% | 10% |
| Profession | Engineer | Plumber | Technician | Electrician | Mechanic |
| Age | 40 years | 55 years | 50 years | 45 years | 35 years |

### 2.7.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Billy | 0 | 1 | 0 | 0 | 0 |
| Blue | 0 | 0 | 1 | 0 | 0 |
| Dennis | 1 | 0 | 0 | 0 | 0 |
| Drill | 0 | 1 | 0 | 0 | 0 |
| Electrician | 0 | 0 | 0 | 1 | 0 |
| Engineer | 1 | 0 | 0 | 0 | 0 |
| FifteenPercent | 1 | 0 | 0 | 0 | 0 |
| Fifty | 0 | 0 | 1 | 0 | 0 |
| FiftyFive | 0 | 1 | 0 | 0 | 0 |
| FivePercent | 0 | 0 | 1 | 0 | 0 |
| Forty | 1 | 0 | 0 | 0 | 0 |
| FortyFive | 0 | 0 | 0 | 1 | 0 |
| George | 0 | 0 | 1 | 0 | 0 |
| Green | 1 | 0 | 0 | 0 | 0 |
| Hammer | 0 | 0 | 1 | 0 | 0 |
| Larry | 0 | 0 | 0 | 1 | 0 |
| Man1 | 1 | 0 | 0 | 0 | 0 |
| Man2 | 0 | 1 | 0 | 0 | 0 |
| Man3 | 0 | 0 | 1 | 0 | 0 |
| Man4 | 0 | 0 | 0 | 1 | 0 |
| Man5 | 0 | 0 | 0 | 0 | 1 |
| Mechanic | 0 | 0 | 0 | 0 | 1 |
| Philip | 0 | 0 | 0 | 0 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| Plumber | 0 | 1 | 0 | 0 | 0 |
| Red | 0 | 1 | 0 | 0 | 0 |
| Saw | 0 | 0 | 0 | 0 | 1 |
| Screwdriver | 0 | 0 | 0 | 1 | 0 |
| Technician | 0 | 0 | 1 | 0 | 0 |
| TenPercent | 0 | 0 | 0 | 0 | 1 |
| ThirtyFive | 0 | 0 | 0 | 0 | 1 |
| TwentyFivePercent | 0 | 0 | 0 | 1 | 0 |
| TwentyPercent | 0 | 1 | 0 | 0 | 0 |
| White | 0 | 0 | 0 | 0 | 1 |
| Wrench | 1 | 0 | 0 | 0 | 0 |
| Yellow | 0 | 0 | 0 | 1 | 0 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| inbetween | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.7.5 Prover9 Results

Proof from prover9, prooving line 1022 by using lines 413, 1018 and 1019.

```
413 -Hammer(m3) | -Hammer(m4).  [resolve(211,c,75,a)].
1018 Hammer(m3).  [resolve(1015,b,999,a),merge(b)].
1019 Hammer(m4).  [back_unit_del(849),unit_del(b,1018)].
1022 $F.  [back_unit_del(413),unit_del(a,1018),unit_del(b,1019)].
```

Back-unit deletion at line 413 $\Longrightarrow$ Copy clause 413.
**P or Q and —Q $\longrightarrow$ P**

Q = —Hammer(m4)
—Q = Hammer(m4)
$\longrightarrow$ **P = —Hammer(m3)**

Unit-deletion at line 1018 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 1018 (which is a unit clause).

Unit-deletion at line 1019 $\Longrightarrow$ The second literal **b** has been removed because it was an instance of the negation clause 1019 (which is a unit clause).

**Q and —Q or R$\longrightarrow$ R**

Q = Hammer(m3)
—Q = —Hammer(m3)
$\longrightarrow$ **R = Hammer(m4)**

## 2.8 Black Friday - Hard Zebra Puzzle

Five friends are side by side drinking juice and talking about the deals they got during the Black Friday sales. Follow the clues to find out who bought the laptop.

### 2.8.1 Clues

1. The man drinking the Orange juice is exactly to the right of the man who got the 70 percent discount.
2. Keith is 45 years old.
3. The man who bought the TV is exactly to the left of the man wearing the Red shirt.
4. At the third position is the man who got the 50 percent discount.
5. Keith is next to the man wearing the White shirt.
6. The 25-year-old man is somewhere between the 35-year-old man and the 40-year-old man, in that order.
7. The man drinking Apple juice bought the Smartphone.
8. The 30-year-old man is exactly to the left of the man that bought the Beard trimmer.
9. Sean is the youngest.
10. The man that got the 40 percent discount is exactly to the right of the man who bought the Beard trimmer.
11. Keith is next to the 35-year-old man.
12. Eugene is 40 years old.
13. Sean is wearing the Black shirt.
14. At the fourth position is the man who got the biggest discount.
15. Dustin got 60 percent off.
16. The man drinking the Lemon juice is exactly to the right of the man drinking the Grape juice.
17. Keith bought a Game console.
18. The man who got the 80 percent discount is exactly to the left of the man who is wearing the Blue shirt.
19. The man drinking Grape juice bought the Beard trimer.
20. The man wearing the Black shirt is somewhere to the right of Keith.
21. The man that bought the Smartphone is next to the man wearing the Black shirt.

### 2.8.2 Human Thinking Solution

1. The fifth man is wearing the blue shirt. **(Clue 18)**
2. The first man has a 10 percent discount, the second man drinks orange juice, the fourth man bought the beard trimmer and is drinking grape juice and the fifth man has a 40 percent discount and is drinking lemon juice. **(Clue 4, Clue 14, Clue 19, Clue 10, Clue 16 and Clue 1)**
3. **Step 2** $\longrightarrow$ The second man can only have the 60 percent discount.
4. **Step 3** $\longrightarrow$ The second man is called Dustin. **(Clue 15)**
5. **Step 2 and Step 4** $\longrightarrow$ The third man bought a smartphone and is drinking apple juice, the fourth man is called Sean, he is 25 years old and is wearing the black shirt and the fifth man is called Eugene and he is 40 years old. **(Clue 21, Clue 7, Clue 9, Clue 13, Clue 16 and Clue 12)**
6. **Step 4, Step 5 and Step 2** $\longrightarrow$ The first man is called Keith, he is 45 years old and he bought a game console. **(Clue 17 and Clue 20)**
7. **Step 6** $\longrightarrow$ The third man can only be named Hank.
8. **Step 6** $\longrightarrow$ The second man is 35 years old. **(Clue 11)**
9. **Step 8** $\longrightarrow$ The third man can only be 30 years old.
10. **Step 6, Step 5 and Step 2** $\longrightarrow$ The second man bought the TV and the third man is wearing the red shirt. **(Clue 3)**
11. **Step 5** $\longrightarrow$ The first man can only be drinking cranberry juice.
12. **Step 10** $\longrightarrow$ The fifth man can only be buying the laptop.
13. **Step 6** $\longrightarrow$ The second man can only be wearing the white shirt. **(Clue 5)**
14. **Step 13** $\longrightarrow$ The first man can only be wearing the green shirt.

### 2.8.3 Mathematical Solution

| | Man #1 | Man #2 | Man #3 | Man #4 | Man #5 |
|---|---|---|---|---|---|
| Shirt | Green | White | Red | Black | Blue |
| Name | Keith | Dustin | Hank | Sean | Eugene |
| Deal | Game Console | TV | Smartphone | Beard Trimmer | Laptop |
| Discount | 70% | 60% | 50% | 80% | 40% |
| Age | 45 years | 35 years | 30 years | 25 years | 40 years |
| Juice | Cranberry | Orange | Apple | Grape | Lemon |

### 2.8.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Apple | 0 | 0 | 1 | 0 | 0 |
| BeardTrimmer | 0 | 0 | 0 | 1 | 0 |
| Black | 0 | 0 | 0 | 1 | 0 |
| Blue | 0 | 0 | 0 | 0 | 1 |
| Cranberry | 1 | 0 | 0 | 0 | 0 |
| Dustin | 0 | 1 | 0 | 0 | 0 |
| EightyPercent | 0 | 0 | 0 | 1 | 0 |
| Eugene | 0 | 0 | 0 | 0 | 1 |
| FiftyPercent | 0 | 0 | 1 | 0 | 0 |
| Forty | 0 | 0 | 0 | 0 | 1 |
| FortyFive | 1 | 0 | 0 | 0 | 0 |
| FortyPercent | 0 | 0 | 0 | 0 | 1 |
| GameConsole | 1 | 0 | 0 | 0 | 0 |
| Grape | 0 | 0 | 0 | 1 | 0 |
| Green | 1 | 0 | 0 | 0 | 0 |
| Hank | 0 | 0 | 1 | 0 | 0 |
| Keith | 1 | 0 | 0 | 0 | 0 |
| Laptop | 0 | 0 | 0 | 0 | 1 |
| Lemon | 0 | 0 | 0 | 0 | 1 |
| Man1 | 1 | 0 | 0 | 0 | 0 |
| Man2 | 0 | 1 | 0 | 0 | 0 |
| Man3 | 0 | 0 | 1 | 0 | 0 |
| Man4 | 0 | 0 | 0 | 1 | 0 |
| Man5 | 0 | 0 | 0 | 0 | 1 |
| Orange | 0 | 1 | 0 | 0 | 0 |
| Red | 0 | 0 | 1 | 0 | 0 |
| Sean | 0 | 0 | 0 | 1 | 0 |
| SeventyPercent | 1 | 0 | 0 | 0 | 0 |

| SixtyPercent | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|
| Smartphone | 0 | 0 | 1 | 0 | 0 |
| TV | 0 | 1 | 0 | 0 | 0 |
| Thirty | 0 | 0 | 1 | 0 | 0 |
| ThirtyFive | 0 | 1 | 0 | 0 | 0 |
| TwentyFive | 0 | 0 | 0 | 1 | 0 |
| White | 0 | 1 | 0 | 0 | 0 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| inbetween | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 1 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.8.5 Prover9 Results

Proof from prover9, prooving line 11693 by using lines 11692, 11352 and 87.

```
87 -directlyright(m2,m2).  [assumption].
11352 BeardTrimmer(m2).  [resolve(11350,b,11337,b),merge(b)].
11692 -BeardTrimmer(x) | directlyright(x,m2).  [resolve(11515,a,233,a)].
11693 $F.  [resolve(11692,a,11352,a),unit_del(a,87)].
```

Resolve at line 11693 $\Longrightarrow$ resolve the first literal of clause 11692 with the first literal of clause 11352.

Unit-deletion at line 87 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 87 (which is a unit clause).

## 2.9 Home Remodeling - Hard Zebra Puzzle

Five couples are side by side buying things to remodel their houses. Each couple chose a room to remodel. Figure out how much they are willing to spend and where they got the inspiration for the remodeling.

### 2.9.1 Clues

1. The couple guiding the Black cart is somewhere between the couple inspired by a Website and the couple guiding the Red cart, in that order.
2. Amelia has a 15,000 dollars budget.
3. The partners who will remodel their Bedroom is next to the couple who will remodel their Living room.
4. Kassie is at one of the ends.
5. Benny is next to the partners steering the Black shopping cart.
6. At the first position is the couple who has the biggest budget.
7. The couple inspired by a Magazine is exactly to the right of the couple that will remodel their Living room.
8. Jeanne is immediately before the partners that are going to remodel their Bathroom.
9. Toni was inspired by a interior design Exhibition.
10. The couple guiding the Green cart is going to remodel their Bathroom.
11. At the fifth position is the couple who got an inspiration from a Friend's house.
12. Benny is next to Sylvia and her husband.
13. Vince is exactly to the right of the couple that will remodel their Kitchen.
14. Myles and his wife will remodel their Bathroom.
15. The couple inspired by a TV show is somewhere between the couple that has the smallest budget and the couple that has a 9,000 dollars budget, in that order.
16. The partners steering the the Red cart is somewhere to the left of the couple steering the White cart.
17. The couple that will remodel their Kitchen is immediately before the couple that will remodel their Bedroom.
18. The partners that have a 12,000 dollars budget is somewhere to the right of the partners guiding the Green cart.
19. Darrel is exactly to the right of the couple who has a 9,000 dollars budget.

### 2.9.2 Human Thinking Solution

1. At the first position, the couple's budget is 15.000 dollars and the wife is Amelia. **(Clue 6 and Clue 2)**
2. The couples that are remodeling the kitchen, bedroom and living room are right next to each other, exactly in this order. **(Clue 17 and Clue 3)**
3. The couple that is remodeling the bathroom cannot be positioned at neither of the ends. **(Clue 8 and Clue 18)**
4. **Step 2 and Step 3** $\longrightarrow$ The couple that is remodeling the bathroom is either the second, or the fourth.
5. **Step 1 and Step 4** $\longrightarrow$ The couple that is remodeling the bathroom can only be the fourth.
6. **Step 2 and Step 5** $\longrightarrow$ The couples that are remodeling the kitchen, bedroom and living room are the first, second and third.
7. **Step 6** $\longrightarrow$ The couple that is remodeling the dining room is the fifth.
8. **Step 7** $\longrightarrow$ Jeanne is carrying is remodeling her bathroom, the couple that is remodeling their bathroom are carrying a green cart and the couple that is remodeling the dining room have a budget of 12.000 dollars. **(Clue 8, Clue 10 and Clue 18)**
9. At the fourth position, the couple was inspired by a magazine, while at the fifth position they were inspired by a friend's house. **(Clue 7 and Clue 11)**
10. **Step 1** $\longrightarrow$ Kassie can only be at the fifth position. **(Clue 4)**
11. **Step 9** $\longrightarrow$ Toni can only be at the second position and her inspiration was an exhibition. **(Clue 9)**
12. **Step 9, Step 8 and Step 11** $\longrightarrow$ The couple inspired by a website can only be at the first position. **(Clue 1)**
13. **Step 12** $\longrightarrow$ The couple inspired by a TV show can only be at the third position.
14. The couples that are carrying the black, red and white carts need to be positioned in this order, but not necessarily next to each other. **(Clue 1 and Clue 16)**
15. **Step 7, Step 15** $\longrightarrow$ The couples that are carrying the black, red and white carts are the second, third and the fifth.
16. **Step 16** $\longrightarrow$ The couple that is carrying the blue cart can only be the first.
17. The couple that has a budget of 3.000 dollars is the second, while the one that has a budget of 9.000 dollars is the fourth. **(Clue 15)**
18. **Step 17** $\longrightarrow$ The couple that has a budget of 6.000 dollars can only be the third.
19. **Step 9** $\longrightarrow$ Myles and Sylvia is at the fourth position. **(Clue 14)**
20. **Step 19** $\longrightarrow$ Benny is the third, while Darrel is the fifth. **(Clue 19)**
21. Vince is the second. **Clue 13**
22. **Step 21** $\longrightarrow$ Stephen is the first.

### 2.9.3 Mathematical Solution

| | Customer #1 | Customer #2 | Customer #3 | Customer #4 | Customer #5 |
|---|---|---|---|---|---|
| Shirt | Blue | Black | Red | Green | White |
| Name | Stephen | Vince | Benny | Myles | Darrel |
| Product | Amelia | Toni | Jeanne | Sylvia | Kassie |
| Discount | Kitchen | Bedroom | Living Room | Bathroom | Dining Room |
| Delivery | 15.000 $ | 3.000 $ | 6.000 $ | 9.000 $ | 12.000 $ |
| Age | Website | Exhibition | TV show | Magazine | Friend's house |

### 2.9.4 Mace4 Results

This puzzle was solved using predicates.

| | | | | | |
|---|---|---|---|---|---|
| Amelia | 1 | 0 | 0 | 0 | 0 |
| Bathroom | 0 | 0 | 0 | 1 | 0 |
| Bedroom | 0 | 1 | 0 | 0 | 0 |
| Benny | 0 | 0 | 1 | 0 | 0 |
| Black | 0 | 1 | 0 | 0 | 0 |
| Blue | 1 | 0 | 0 | 0 | 0 |
| Couple1 | 1 | 0 | 0 | 0 | 0 |
| Couple2 | 0 | 1 | 0 | 0 | 0 |
| Couple3 | 0 | 0 | 1 | 0 | 0 |
| Couple4 | 0 | 0 | 0 | 1 | 0 |
| Couple5 | 0 | 0 | 0 | 0 | 1 |
| Darrel | 0 | 0 | 0 | 0 | 1 |
| DiningRoom | 0 | 0 | 0 | 0 | 1 |
| Exhibition | 0 | 1 | 0 | 0 | 0 |
| FifteenThousand | 1 | 0 | 0 | 0 | 0 |
| FriendsHouse | 0 | 0 | 0 | 0 | 1 |
| Green | 0 | 0 | 0 | 1 | 0 |
| Jeanne | 0 | 0 | 1 | 0 | 0 |
| Kassie | 0 | 0 | 0 | 0 | 1 |
| Kitchen | 1 | 0 | 0 | 0 | 0 |
| LivingRoom | 0 | 0 | 1 | 0 | 0 |
| Magazine | 0 | 0 | 0 | 1 | 0 |
| Myles | 0 | 0 | 0 | 1 | 0 |
| NineThousand | 0 | 0 | 0 | 1 | 0 |
| Red | 0 | 0 | 1 | 0 | 0 |
| SixThousand | 0 | 0 | 1 | 0 | 0 |
| Stephen | 1 | 0 | 0 | 0 | 0 |
| Sylvia | 0 | 0 | 1 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| TVshow | 0 | 0 | 1 | 0 | 0 |
| ThreeThousand | 0 | 1 | 0 | 0 | 0 |
| Toni | 0 | 1 | 0 | 0 | 0 |
| TwelveThousand | 0 | 1 | 0 | 0 | 0 |
| Vince | 0 | 1 | 0 | 0 | 0 |
| Website | 1 | 0 | 0 | 0 | 0 |
| White | 0 | 0 | 0 | 0 | 1 |
| extremity | 1 | 0 | 0 | 0 | 1 |

| differentFrom | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 |

| directlyright | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

| nextto | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 | 0 |

| right | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

### 2.9.5 Prover9 Results

Proof from prover9, prooving line 11693 by using lines 11692, 11352 and 87.

```
14881 Bathroom(d4) | Bathroom(d2) | Bathroom(d3).  [back_unit_del(11605),
                                                   unit_del(d,14880)].
14882 -Bathroom(d2).  [back_unit_del(493),unit_del(a,14880)].
14883 -Bathroom(d3).  [back_unit_del(492),unit_del(a,14880)].
14884 -Bathroom(d4).  [back_unit_del(491),unit_del(a,14880)].
14886 $F.  [back_unit_del(14881),unit_del(a,14884),unit_del(b,14882),
            unit_del(c,14883)].
```

Back-unit deletion at line 14881 $\Longrightarrow$ Copy clause 14881.

Unit-deletion at line 14884 $\Longrightarrow$ The first literal **a** has been removed because it was an instance of the negation clause 14884 (which is a unit clause).

Unit-deletion at line 14882 $\Longrightarrow$ The second literal **b** has been removed because it was an instance of the negation clause 14882 (which is a unit clause).

Unit-deletion at line 14883 $\Longrightarrow$ The third literal **c** has been removed because it was an instance of the negation clause 14883 (which is a unit clause).

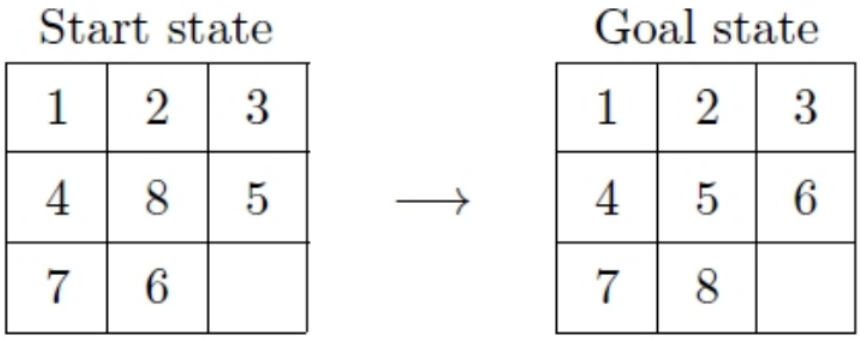
# Chapter 3

# A3: Planning

Each one of the exercises was solved using the FD - Fast Downward Planning System. Given that the planning of uncertain domains is required and observation actions were used, Contingent Planning was the obvious choice.

## 3.1 Sliding Tiles

In a 3 x 3 matrix, there are eight numbered tiles (from 1 to 8) and one blank tile. The goal is to shuffle the numbered tiles into the right order.




### 3.1.1 Domain

The domain contains tiles that should be slided from a start position to a goal one, as depicted in the figure from above. The **sliding-tile** domain contains four predicates and one action. The position of the tile is formalised by predicate **(tile-at ?tile ?row ?col)**. Blank position is encoded by **(is-blank ?c1 ?c2)**. The **move-tile-down** action has four parameters: the **?tile**, which is moved from the current row **?old-row** to the next row **?new-row**, and the column **?col**. the preconditions check if the new position is blank. In that case, two facts are removed from the current state (current tile position and current blank and two new facts are added (the new tile position and the new blank).

Here's the domain's code below:

```
(define (domain sliding-tile)
(:predicates
(tile-at ?t ?r ?c)
(is-blank ?r ?c)
(next-row ?r1 ?r2)
(next-col ?c1 ?c2))
(:action move-tile-down
:parameters (?tile ?old-row ?new-row ?col)
:precondition (and (next-row ?old-row ?new-row)
    (tile-at ?tile ?old-row ?col)
    (is-blank ?new-row ?col))
:effect        (and (not (tile-at ?tile ?old-row ?col))
     (not (is-blank ?new-row ?col))
     (tile-at ?tile ?new-row ?col)
     (is-blank ?old-row ?col)))
(:action move-tile-up
  :parameters(?tile ?old-row ?new-row ?col)
  :precondition (and (next-row ?new-row ?old-row)
```

```
      (tile-at ?tile ?old-row ?col)
       (is-blank ?new-row ?col))
  :effect(and (not (tile-at ?tile ?old-row ?col))
       (not (is-blank ?new-row ?col))
       (tile-at ?tile ?new-row ?col)
       (is-blank ?old-row ?col)))
(:action move-tile-right
  :parameters(?tile ?row ?old-col ?new-col)
  :precondition (and (next-col ?old-col ?new-col)
      (tile-at ?tile ?row ?old-col)
       (is-blank ?row ?new-col))
  :effect(and (not (tile-at ?tile ?row ?old-col))
       (not (is-blank ?row ?new-col))
       (tile-at ?tile ?row ?new-col)
       (is-blank ?row ?old-col)))
(:action move-tile-left
  :parameters(?tile ?row ?old-col ?new-col)
  :precondition (and (next-col ?new-col ?old-col)
      (tile-at ?tile ?row ?old-col)
       (is-blank ?row ?new-col))
  :effect(and (not (tile-at ?tile ?row ?old-col))
       (not (is-blank ?row ?new-col))
       (tile-at ?tile ?row ?new-col)
       (is-blank ?row ?old-col))))
```

### 3.1.2 Problem

There are eight object tiles, three object rows and three object columns. The arrangement of the tiles, rows and columns are set as depicted in the figure from the previous page, as well as the goal state.

Here's the problem's code below:

```
(define (problem eight-puzzle)
(:domain sliding-tile)
(:objects
tile1 tile2 tile3 tile4 tile5 tile6 tile7 tile8
row1 row2 row3
col1 col2 col3)
(:init
(next-row row1 row2)       (next-col col1 col2)
(next-row row2 row3)       (next-col col2 col3)
(tile-at tile8 row2 col2)  (tile-at tile4 row2 col1)
(tile-at tile7 row3 col1)  (tile-at tile3 row1 col3)
(tile-at tile6 row3 col2)  (tile-at tile2 row1 col2)
(tile-at tile5 row2 col3)  (tile-at tile1 row1 col1)
(is-blank row3 col3))
(:goal
   (and
    (tile-at tile1 row1 col1)  (tile-at tile2 row1 col2)
```

```
  (tile-at tile3 row1 col3)  (tile-at tile4 row2 col1)
  (tile-at tile5 row2 col2)  (tile-at tile6 row2 col3)
  (tile-at tile7 row3 col1)  (tile-at tile8 row3 col2))))
```

### 3.1.3 Results

Run command:

```
./Contingent-FF -o ./solved/sliding-tiles/sliding-tiles-domain.pddl -f ./
solved/sliding-tiles/sliding-tiles-problem.pddl
```

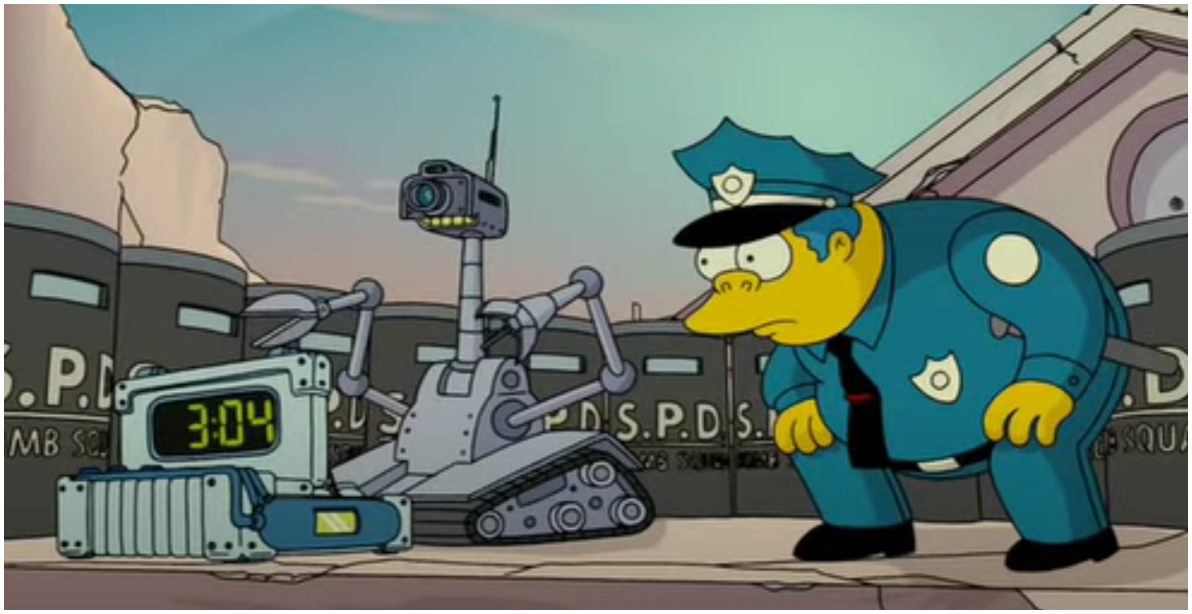
Terminal output:

```
ff: found plan as follows
-------------------------------------------------
  0||0 --- MOVE-TILE-RIGHT TILE6 ROW3 COL2 COL3 --- SON: 1||0
-------------------------------------------------
  1||0 --- MOVE-TILE-DOWN TILE8 ROW2 ROW3 COL2 --- SON: 2||0
-------------------------------------------------
  2||0 --- MOVE-TILE-LEFT TILE5 ROW2 COL3 COL2 --- SON: 3||0
-------------------------------------------------
  3||0 --- MOVE-TILE-UP TILE6 ROW3 ROW2 COL3 --- SON: 4||-1
-------------------------------------------------

tree layers: 4
total nr. actions: 4
```

*The cost of the plan is given by the number of layers (4).

## 3.2 Packages with bombs

A robot is given several packages **pi**, and told that some of them may contain bombs **bi**. The robot's goal is to find the bombs and defuse them.




### 3.2.1 Domain

First of all, there are three types of objects: **package**, **bomb** and **toilet**.

Second of all, there are defined four predicates: **in** [package type object] [bomb type object] (which specifies if there is a bomb type object in the package type object, or not), **defused** [bomb type object] (which specifies if the bomb type object is defused, or not), **clog** [toilet type object] (which specifies if the toilet type object is clogged, or not) and **stuck** [toilet type object] (which specifies if the toilet type object is stuck, or not).

Moreover, there are four types of actions that can be taken: **senseIN**, **dunk**, **flush** and **unstick**. The only way to **defuse** a bomb is for the robot to **dunk** the package containing the bomb in the toilet. Placing a package in the toilet might **clog** the toilet. The robot can sense if a bomb is within a package by using the observation action **senseIN**. Action **dunk** takes place only when the bomb is in the package. Important here is the nondeterministic effect **(stuck ?t)**, signaled by the keyword **nondet**. Without preconditions, action **flush** can be executed at each time instance with the certain effect to unclog the toilet **(not (clog ?t))**. Action **unstick** can also be executed anytime, but its effect is conditioned by the state variable **(stuck ?toilet)**.

Here's the domain's code below:

```
(define (domain btnd)
    (:types package bomb toilet)
    (:predicates
        (in ?p - package ?b - bomb)
        (defused ?b - bomb)
        (clog ?toilet - toilet)
```

```
        (stuck ?toilet - toilet))

    (:action senseIN
        :parameters (?p - package ?b - bomb)
        :observe (in ?p ?b))

    (:action dunk
        :parameters (?p - package ?b - bomb ?t - toilet)
        :precondition (and (not (clog ?t)) (not (stuck ?t)))
        :effect (and (when (in ?p ?b) (defused ?b))
            (clog ?t)
            (nondet (stuck ?t))))

    (:action flush
        :parameters (?t - toilet)
        :effect (not (clog ?t)))

    (:action unstick
        :parameters (?toilet - toilet)
        :effect (and (when (stuck ?toilet) (not (stuck ?toilet))))))
```

### 3.2.2 Problem

There are two bombs, four packages and one toilet. We know that bomb **b0** is in package **p0**. We also know that **b1** is in one of the remaining three packages **p1**, **p2** or **p3**. The goal is to defuse the bombs.

Here's the problem's code below:

```
(define (problem btnd - 4)
    (:domain btnd)
    (:objects
        b0 b1 - bomb
        p0 p1 p2 p3 - package
        t0 - toilet)

    (:init
        (in p0 b0)
        (unknown (in p1 b1))
        (unknown (in p2 b1))
        (unknown (in p3 b1))
        (oneof (in p1 b1) (in p2 b1) (in p3 b1)))
    (:goal (and (defused b0) (defused b1))))
```

### 3.2.3 Results

Run command:

```
./Contingent-FF -o ./solved/packages-with-bombs/packages-with-bombs-domain.
pddl -f ./solved/packages-with-bombs/packages-with-bombs-problem.pddl
```

Terminal output:

```
ff: found plan as follows
------------------------------------------------
  0||0 --- DUNK P3 B1 T0 --- SON: 1||0
------------------------------------------------
  1||0 --- FLUSH T0 --- SON: 2||0
------------------------------------------------
  2||0 --- UNSTICK T0 --- SON: 3||0
------------------------------------------------
  3||0 --- DUNK P0 B0 T0 --- SON: 4||0
------------------------------------------------
  4||0 --- UNSTICK T0 --- SON: 5||0
------------------------------------------------
  5||0 --- FLUSH T0 --- SON: 6||0
------------------------------------------------
  6||0 --- DUNK P2 B1 T0 --- SON: 7||0
------------------------------------------------
  7||0 --- UNSTICK T0 --- SON: 8||0
------------------------------------------------
  8||0 --- FLUSH T0 --- SON: 9||0
------------------------------------------------
  9||0 --- DUNK P1 B1 T0 --- SON: 10||-1
------------------------------------------------

tree layers: 10
total nr. actions: 10
```

*The cost of the plan is given by the number of layers (10).
*The found plan is a linear plan for the nondeterministic bomb domain.

## 3.3 Clinical Trial

There are four volunteers that want to enroll into a new clinical trial research. In order to be selected into the trial, the volunteers must be diagnosed as healthy after being tested. If they want, any volunteer can be additionally tested.

### 3.3.1 Domain

There are defined five predicates for each object: **is-candidate** (which specifies if the object is a candidate, or not), **healthy** (which specifies if the object is a healthy, or not), **blood-tested** (which specifies if the object has been through the first set of tests, or not), **is-selected** (which specifies if the object is selected, or not) and **additional-tested** (which specifies if the object needs additional testing, or not).

Moreover, there are five types of actions that can be taken: **senseADDITIONALTESTS** (which specifies if an object has been previously additionally tested, or not), **make-blood-tests** (which specifies if an object can be blood tested, or not), **make-more-tests** (which specifies if an object can be additionally tested, or not), **select-candidate** (which specifies if an object can be selected, or not) and **give-medication** (which specifies if an object needs to be given medication after the blood test, or not).

More details about the actions.
Here's the domain's code below:

```
(define (domain clinical-trial)
(:predicates (is-candidate ?p)
             (healthy ?p)
             (blood-tested ?p)
             (is-selected ?p)
             (additional-tested ?p))
```

```
    (:action senseADDITIONALTESTS
    :parameters (?p)
    :observe (additional-tested ?p))

    (:action make-blood-tests
    :parameters (?p)
    :precondition (and (is-candidate ?p) (additional-tested ?p))
    :effect (and (nondet (healthy ?p)) (blood-tested ?p)))

    (:action make-more-tests
    :parameters (?p)
    :precondition (and (is-candidate ?p) (not (additional-tested ?p)))
    :effect (and (additional-tested ?p)))

    (:action select-candidate
    :parameters (?p)
    :precondition (and (healthy ?p) (is-candidate ?p) (blood-tested ?p))
    :effect (and (is-selected ?p)))

    (:action give-medication
    :parameters (?p)
    :precondition (and (blood-tested ?p))
    :effect (and (when (not (healthy ?p)) (healthy ?p))))
)
```

### 3.3.2 Problem

There are defined four candidate objects. The predicate **is-candidate** is set for every object (because all of the objects are candidates). Only candidates number three and four have stated that they need additional tests, but this matter is **unknown** for candidates number one and two (it is only know that at least one of them will be getting additional tests). The goal is that all of the candidates will be selected for the trial.

Here's the problem's code below:

```
(define (problem clinical-trial-selecting-phase)
    (:domain clinical-trial)
    (:objects candidate1 candidate2 candidate3 candidate4)
    (:init
        (is-candidate candidate1)
        (is-candidate candidate2)
        (is-candidate candidate3)
        (is-candidate candidate4)

        (additional-tested candidate3)
        (additional-tested candidate4)

        (unknown (additional-tested candidate1))
        (unknown (additional-tested candidate2))
```

```
    (oneof (additional-tested candidate1)
           (additional-tested candidate2)))

  (:goal (and (is-selected candidate1) (is-selected candidate3)
           (is-selected candidate4) (is-selected candidate2))))
```

### 3.3.3 Results

Run command:

```
./Contingent-FF -o ./solved/clinical-trial/clinical-trial-domain.pddl -f
./solved/clinical-trial/clinical-trial-problem.pddl
```

Terminal output:

```
ff: found plan as follows
-------------------------------------------------
  0||0 --- MAKE-BLOOD-TESTS CANDIDATE4 --- SON: 1||0
-------------------------------------------------
  1||0 --- MAKE-BLOOD-TESTS CANDIDATE3 --- SON: 2||0
-------------------------------------------------
  2||0 --- GIVE-MEDICATION CANDIDATE3 --- SON: 3||0
-------------------------------------------------
  3||0 --- SELECT-CANDIDATE CANDIDATE3 --- SON: 4||0
-------------------------------------------------
  4||0 --- GIVE-MEDICATION CANDIDATE4 --- SON: 5||0
-------------------------------------------------
  5||0 --- SELECT-CANDIDATE CANDIDATE4 --- SON: 6||0
-------------------------------------------------
  6||0 --- SENSEADDITIONALTESTS CANDIDATE2 --- TRUESON: 7||0 --- FALSESON:
                                                                    7||1
-------------------------------------------------
  7||0 --- MAKE-BLOOD-TESTS CANDIDATE2 --- SON: 8||0
  7||1 --- MAKE-MORE-TESTS CANDIDATE2 --- SON: 8||1
-------------------------------------------------
  8||0 --- MAKE-MORE-TESTS CANDIDATE1 --- SON: 9||0
  8||1 --- MAKE-BLOOD-TESTS CANDIDATE2 --- SON: 9||1
-------------------------------------------------
  9||0 --- MAKE-BLOOD-TESTS CANDIDATE1 --- SON: 10||0
  9||1 --- MAKE-BLOOD-TESTS CANDIDATE1 --- SON: 10||1
-------------------------------------------------
 10||0 --- GIVE-MEDICATION CANDIDATE1 --- SON: 11||0
 10||1 --- GIVE-MEDICATION CANDIDATE1 --- SON: 11||1
-------------------------------------------------
 11||0 --- SELECT-CANDIDATE CANDIDATE1 --- SON: 12||0
 11||1 --- SELECT-CANDIDATE CANDIDATE1 --- SON: 12||1
-------------------------------------------------
 12||0 --- GIVE-MEDICATION CANDIDATE2 --- SON: 13||0
 12||1 --- GIVE-MEDICATION CANDIDATE2 --- SON: 13||1
-------------------------------------------------
```

```
 13||0 --- SELECT-CANDIDATE CANDIDATE2 --- SON: 14||-1
 13||1 --- SELECT-CANDIDATE CANDIDATE2 --- SON: 14||-1
-----------------------------------------------

tree layers: 14
total nr. actions: 21
```

*The cost of the plan is given by the number of layers (14).

# Bibliography

# Appendix A

# Your original code

## A.1 Stoian Iulia Tia's code:

### A.1.1 Search

Code starts from the next page.

```
import display
import structures
from Queue import *
from copy import deepcopy

def isLegalMove(tempx, tempy):
    if tempx < 0 or tempx > 2:
        return False

    if tempy < 0 or tempy > 2:
        return False

    return True

def switchMove(index):
    switcher = {
        0: "N",
        1: "E",
        2: "S",
        3: "W"
    }

    return switcher.get(index)

class PuzzleModel:

    def __init__(self, puzzle):
        self.puzzle = puzzle
        self.startState = structures.Node(puzzle) #puzzle = array de int uri

    def solveFunction(self):
        print("solving shit")

        #print("Considering BFS: ", self.BFS())
        print("Considering DFS: ", self.IDDFS())
        #print("Considering Greedy with Manhattan Distance Heuristic",
        self.greedySearch(0))
        #print("Considering Greedy with Euclidian Distance Heuristic",
        self.greedySearch(1))
        #print("Considering A* with Manhattan Distance Heuristic", self.aStarSearch(0))
        #print("Considering A* with Euclidian Distance Heuristic", self.aStarSearch(1))

    def isGoalState(self, currentState):
        goalState = [0, 1, 2, 3, 4, 5, 6, 7, 8]

        #print(currentState.puzzle)
        #print(goalState)

        for i in range(0, 9):
           if (currentState.puzzle[i] != goalState[i]):
                #print("The current state is not the goal state.")
                return False

        #print("The current state is the goal state.")
        return True

    def getSuccessors(self, currentState):
        x = y = tile = 0

        for i in range(0, 9):
            if currentState.puzzle[i] == 0:
                x = i / 3
                y = i % 3
                tile = i
                break

        dx = [-1, 0, 1, 0]
```

```
66            dy = [0, 1, 0, -1]
67
68            successors = Queue()
69            for i in range(0, 4):
70                tempx = x + dx[i]
71                tempy = y + dy[i]
72
73                if isLegalMove(tempx, tempy) == True:
74                    newPuzzle = deepcopy(currentState.puzzle)
75                    newTile = tempx * 3 + tempy % 3
76
77                    newPuzzle[tile] = newPuzzle[newTile]
78                    newPuzzle[newTile] = 0
79
80                    action = switchMove(i)
81                    #print(action)
82
83                    newState = structures.Node(newPuzzle, currentState, action)
84                    successors.put(newState)
85
86            return successors
87
88        def BFS(self):
89
90            frontier = Queue()
91            steps = list()
92
93            frontier.put(self.startState)
94            while True:
95                if frontier.empty():
96                    return None # nu gaseste solutie daca vreodata frontiera ajunge goala
97
98                currentState = frontier.get()
99                if self.isGoalState(currentState):
100                   return currentState.actions
101
102               elif currentState not in steps:
103                   steps.append(currentState)
104                   successors = self.getSuccessors(currentState)
105                   while not successors.empty():
106                       frontier.put(successors.get())
107
108       # Iterative Deepening Depth-First Search Algorithm
109       def IDDFS(self):
110           depth = 0
111           currentState = None
112           while currentState == None:
113               currentState = self.DLS(depth)
114               depth = depth + 1
115
116           return currentState.actions
117
118       # Depth Limited Search ALgorithm
119       def DLS(self, depth):
120           frontier = LifoQueue()
121           frontier.put(self.startState)
122
123           while True:
124               if frontier.empty():
125                   return None
126
127               currentState = frontier.get()
128               if self.isGoalState(currentState):
129                   return currentState
130
131               elif currentState.depth != depth:
132                   successors = self.getSuccessors(currentState)
```

```
133                 while not successors.empty():
134                     frontier.put(successors.get())
135
136     # nu merge
137     def greedySearch(self, type):
138         currentState = self.startState
139
140         frontier = PriorityQueue()
141         steps = list()
142
143         frontier.put((currentState.heuristic(type), currentState))
144
145         while True:
146             if frontier.empty():
147                 return None
148
149             currentState = frontier.get()[1]
150
151             print(currentState.puzzle)
152
153             if self.isGoalState(currentState):
154                 print(currentState.actions)
155                 return currentState
156
157             elif currentState not in steps:
158                 steps.append(currentState)
159                 successors = self.getSuccessors(currentState)
160
161                 while not successors.empty():
162                     successor = successors.get()
163                     frontier.put((successor.heuristic(type), successor))
164
165     def aStarSearch(self, type):
166         currentState = self.startState
167         frontier = PriorityQueue()
168         frontier.put((currentState.heuristic(type), currentState))
169         steps = list()
170
171         while True:
172             if frontier.empty():
173                 return None
174
175             currentState = frontier.get()[1]
176             if self.isGoalState(currentState):
177                 print(currentState.actions)
178                 return currentState
179
180             elif currentState not in steps:
181                 steps.append(currentState)
182                 successors = self.getSuccessors(currentState)
183
184                 while not successors.empty():
185                     successor = successors.get()
186                     frontier.put((successor.heuristic(type) + successor.depth,
                        successor))
```

### A.1.2 Logics

**Who owns the crocodile?** (page 1 of 4)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Girl3(g3) -> Crocodile(g3).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Girls' order in row
    % Girl1: first girl
    % Girl2: second girl
    % Girl3: third girl
    % Girl4: fourth girl
    % Girl5: fifth girl

% Girls' names
    % Georgina, Hannah, Jessica, Jo, Lucy

% Favourite Colour
    % Blue, Green, Lilac, Pink, Purple

% Favourite Chocolate
    % Boost, Crunchies, DairyMilk: Dairy Millk, MilkyBars: Milky Bars, WispaBites:
    Wispa Bites

% Favourite Pet
    % Crocodile, Hamster, Horse, Puppy, Rabbit

% Favourite Hobby
    % Riding: Horse riding, Skiing, Bowling: Ten-Pin bowling, Tennis

% Favourite Holiday
    % Australia, Canada, Florida, Maldives, Tobago

formulas(assumptions).

% The 5 girls differ
    differentFrom(g1, g2).
    differentFrom(g1, g3).
    differentFrom(g1, g4).
    differentFrom(g1, g5).
    differentFrom(g2, g3).
    differentFrom(g2, g4).
    differentFrom(g2, g5).
    differentFrom(g3, g4).
    differentFrom(g3, g5).
    differentFrom(g4, g5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    g1 = 0.
    g2 = 1.
    g3 = 2.
    g4 = 3.
    g5 = 4.

    Girl1(0).
    Girl2(1).
    Girl3(2).
    Girl4(3).
    Girl5(4).

```

**Who owns the crocodile?** (page 2 of 4)

```
% Relation directlyright(x, y) that affirms that "girl y" is directly to the right of "girl x"
    directlyright(g1, g2).
    directlyright(g2, g3).
    directlyright(g3, g4).
    directlyright(g4, g5).

    -directlyright(g1, g1).
    -directlyright(g2, g2).
    -directlyright(g3, g3).
    -directlyright(g4, g4).
    -directlyright(g5, g5).
    -directlyright(g1, g3).
    -directlyright(g2, g4).
    -directlyright(g3, g5).
    -directlyright(g3, g1).
    -directlyright(g4, g2).
    -directlyright(g5, g3).
    -directlyright(g2, g1).
    -directlyright(g3, g2).
    -directlyright(g4, g3).
    -directlyright(g5, g4).
    -directlyright(g1, g4).
    -directlyright(g2, g5).
    -directlyright(g4, g1).
    -directlyright(g5, g2).
    -directlyright(g1, g5).
    -directlyright(g5, g1).

% Relation nextto(x, y) that affirms that "girl y" is next to "girl x"
    nextto(g1, g2).
    nextto(g2, g1).
    nextto(g2, g3).
    nextto(g3, g2).
    nextto(g3, g4).
    nextto(g4, g3).
    nextto(g4, g5).
    nextto(g5, g4).

    -nextto(g1, g1).
    -nextto(g2, g2).
    -nextto(g3, g3).
    -nextto(g4, g4).
    -nextto(g5, g5).
    -nextto(g1, g3).
    -nextto(g2, g4).
    -nextto(g3, g5).
    -nextto(g3, g1).
    -nextto(g4, g2).
    -nextto(g5, g3).
    -nextto(g1, g4).
    -nextto(g2, g5).
    -nextto(g4, g1).
    -nextto(g5, g2).
    -nextto(g1, g5).
    -nextto(g5, g1).

% Each girl is different and has a name, favourite colour, chocolate, pet, hobby and holiday
    Girl1(x) | Girl2(x) | Girl3(x) | Girl4(x) | Girl5(x).
    Georgina(x) | Hannah(x) | Jessica(x) | Jo(x) | Lucy(x).
    Blue(x) | Green(x) | Lilac(x) | Pink(x) | Purple(x).
    Boost(x) | Crunchies(x) | DairyMilk(x) | MilkyBars(x) | WispaBites(x).
    Crocodile(x) | Hamster(x) | Horse(x) | Puppy(x) | Rabbit(x).
    Riding(x) | Skiing(x) | Swimming(x) | Bowling(x) | Tennis(x).
    Australia(x) | Canada(x) | Florida(x) | Maldives(x) | Tobago(x).

% Each property applies at most to one girl.
    Georgina(x) & Georgina(y) -> -differentFrom(x, y).
```

```
136       Hannah(x) & Hannah(y) -> -differentFrom(x, y).
137       Jessica(x) & Jessica(y) -> -differentFrom(x, y).
138       Jo(x) & Jo(y) -> -differentFrom(x, y).
139       Lucy(x) & Lucy(y) -> -differentFrom(x, y).
140
141       Blue(x) & Blue(y) -> -differentFrom(x, y).
142       Green(x) & Green(y) -> -differentFrom(x, y).
143       Lilac(x) & Lilac(y) -> -differentFrom(x, y).
144       Pink(x) & Pink(y) -> -differentFrom(x, y).
145       Purple(x) & Purple(y) -> -differentFrom(x, y).
146
147       Boost(x) & Boost(y) -> -differentFrom(x, y).
148       Crunchies(x) & Crunchies(y) -> -differentFrom(x, y).
149       DairyMilk(x) & DairyMilk(y) -> -differentFrom(x, y).
150       MilkyBars(x) & MilkyBars(y) -> -differentFrom(x, y).
151       WispaBites(x) & WispaBites(y) -> -differentFrom(x, y).
152
153       Crocodile(x) & Crocodile(y) -> -differentFrom(x, y).
154       Hamster(x) & Hamster(y) -> -differentFrom(x, y).
155       Horse(x) & Horse(y) -> -differentFrom(x, y).
156       Puppy(x) & Puppy(y) -> -differentFrom(x, y).
157       Rabbit(x) & Rabbit(y) -> -differentFrom(x, y).
158
159       Riding(x) & Riding(y) -> -differentFrom(x, y).
160       Skiing(x) & Skiing(y) -> -differentFrom(x, y).
161       Swimming(x) & Swimming(y) -> -differentFrom(x, y).
162       Bowling(x) & Bowling(y) -> -differentFrom(x, y).
163       Tennis(x) & Tennis(y) -> -differentFrom(x, y).
164
165       Australia(x) & Australia(y) -> -differentFrom(x, y).
166       Canada(x) & Canada(y) -> -differentFrom(x, y).
167       Florida(x) & Florida(y) -> -differentFrom(x, y).
168       Maldives(x) & Maldives(y) -> -differentFrom(x, y).
169       Tobago(x) & Tobago(y) -> -differentFrom(x, y).
170
171   %Clues
172
173       % 1. Jo likes the Wispa Bites.
174       Jo(x) -> WispaBites(x).
175
176       % 2. The person with the Hamster likes Swimming.
177       Hamster(x) -> Swimming(x).
178
179       % 3. Hannah eats Dairy Milk.
180       Hannah(x) -> DairyMilk(x).
181
182       % 4. Jessica is on the left of Georgina.
183       Jessica(x) & Georgina(y) -> directlyright(x, y).
184
185       % 5. Lucy is the first on the left.
186       Girl1(x) -> Lucy(x).
187
188       % 6. The first person on the right likes Swimming.
189       Girl5(x) -> Swimming(x).
190
191       % 7. The person who eats Milky Bars owns a Horse.
192       MilkyBars(x) -> Horse(x).
193
194       % 8. The person in the middle eats Dairy Milk.
195       Girl3(x) -> DairyMilk(x).
196
197       % 9. Jessica likes Green.
198       Jessica(x) -> Green(x).
199
200       % 10. The person on the left of the middle wants to go to Tobago.
201       Girl2(x) -> Tobago(x).
202
203       % 11. The person who wants to go to Maldives likes Lilac.
204       Maldives(x) -> Lilac(x).
```

```
205
206       % 12. The person who likes Wispa Bites sits next to the person who wants to go to
          Florida.
207       WispaBites(x) & Florida(y) -> nextto(y, x).
208
209       % 13. The person who likes Pink wants to go to Florida.
210       Pink(x) -> Florida(x).
211
212       % 14. The person who sits first on the left likes lilac.
213       Girl1(x) -> Lilac(x).
214
215       % 15. The girl that likes Blue owns a Puppy.
216       Blue(x) -> Puppy(x).
217
218       % 16. The person who likes Skiing sits next to the person who has a Hamster.
219       Skiing(x) & Hamster(y) -> nextto(y, x).
220
221       % 17. The girl on the right of the girl who likes Tennis likes Horse riding.
222       Tennis(x) & Riding(y) -> directlyright(x, y).
223
224       % 18. The girl next to the girl who likes Milky bars likes Boost.
225       MilkyBars(x) & Boost(y) -> nextto(x, y).
226
227       % 19. The girl who likes Purple wants to go to Canada.
228       Purple(x) -> Canada(x).
229
230       % 20. The girl who likes Crunchies owns a rabbit.
231       Crunchies(x) -> Rabbit(x).
232
233       % 21. The girl who likes Skiing sits next to the girl who play Ten-Pin bowlings.
234       Skiing(x) & Bowling(y) -> nextto(y, x).
235
236       % 22. Jessica wants to go to Australia.
237       Jessica(x) -> Australia(x).
238
239   end_of_list.
240
```

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Girl1(g1) -> Blue(g1).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Girls' order in row
    % Girl1: first girl
    % Girl2: second girl
    % Girl3: third girl
    % Girl4: fourth girl
    % Girl5: fifth girl

% Girls' names
    % Ella, Julia, Lily, Makayla, Riley

% Shirt
    % Black, Blue, Green, Red, White

% Civilization
    % Aztec, Chinese, Egyptian, Greek, Roman

% Subject
    % Architecture, Art, Economy, Religion, Science

% Age
    % Ten, Eleven, Twelve, Thirteen, Fourteen

% Juice
    % Apple, Cranberry, Grapefruit, Orange, Strawberry

formulas(assumptions).

% The 5 girls differ
    differentFrom(g1, g2).
    differentFrom(g1, g3).
    differentFrom(g1, g4).
    differentFrom(g1, g5).
    differentFrom(g2, g3).
    differentFrom(g2, g4).
    differentFrom(g2, g5).
    differentFrom(g3, g4).
    differentFrom(g3, g5).
    differentFrom(g4, g5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    g1 = 0.
    g2 = 1.
    g3 = 2.
    g4 = 3.
    g5 = 4.

    Girl1(0).
    Girl2(1).
    Girl3(2).
    Girl4(3).
    Girl5(4).

% Relation directlyright(x, y) that affirms that "girl y" is directly to the right of
```

```
    "girl x"
70      directlyright(g1, g2).
71      directlyright(g2, g3).
72      directlyright(g3, g4).
73      directlyright(g4, g5).
74
75      -directlyright(g1, g1).
76      -directlyright(g2, g2).
77      -directlyright(g3, g3).
78      -directlyright(g4, g4).
79      -directlyright(g5, g5).
80      -directlyright(g1, g3).
81      -directlyright(g2, g4).
82      -directlyright(g3, g5).
83      -directlyright(g3, g1).
84      -directlyright(g4, g2).
85      -directlyright(g5, g3).
86      -directlyright(g2, g1).
87      -directlyright(g3, g2).
88      -directlyright(g4, g3).
89      -directlyright(g5, g4).
90      -directlyright(g1, g4).
91      -directlyright(g2, g5).
92      -directlyright(g4, g1).
93      -directlyright(g5, g2).
94      -directlyright(g1, g5).
95      -directlyright(g5, g1).
96
97  % Relation nextto(x, y) that affirms that "girl y" is next to "girl x"
98      nextto(g1, g2).
99      nextto(g2, g1).
100     nextto(g2, g3).
101     nextto(g3, g2).
102     nextto(g3, g4).
103     nextto(g4, g3).
104     nextto(g4, g5).
105     nextto(g5, g4).
106
107     -nextto(g1, g1).
108     -nextto(g2, g2).
109     -nextto(g3, g3).
110     -nextto(g4, g4).
111     -nextto(g5, g5).
112     -nextto(g1, g3).
113     -nextto(g2, g4).
114     -nextto(g3, g5).
115     -nextto(g3, g1).
116     -nextto(g4, g2).
117     -nextto(g5, g3).
118     -nextto(g1, g4).
119     -nextto(g2, g5).
120     -nextto(g4, g1).
121     -nextto(g5, g2).
122     -nextto(g1, g5).
123     -nextto(g5, g1).
124
125 % Relation extremity(x) that affirms that "girl x" is at one of the ends
126     extremity(g1).
127     extremity(g5).
128
129     -extremity(g2).
130     -extremity(g3).
131     -extremity(g4).
132
133 % Relation inbetween(x, y) that affirms that there is a girl somewhere in between "girl
    x" and "girl y", exactly in that order
134     inbetween(g1, g3).
135     inbetween(g2, g4).
136     inbetween(g3, g5).
```

```
    inbetween(g1, g4).
    inbetween(g2, g5).
    inbetween(g1, g5).

    -inbetween(g1, g2).
    -inbetween(g2, g1).
    -inbetween(g2, g3).
    -inbetween(g3, g2).
    -inbetween(g3, g4).
    -inbetween(g4, g3).
    -inbetween(g4, g5).
    -inbetween(g5, g4).
    -inbetween(g1, g1).
    -inbetween(g2, g2).
    -inbetween(g3, g3).
    -inbetween(g4, g4).
    -inbetween(g5, g5).
    -inbetween(g3, g1).
    -inbetween(g4, g2).
    -inbetween(g5, g3).
    -inbetween(g4, g1).
    -inbetween(g5, g2).
    -inbetween(g5, g1).

% Relation right(x, y) that affirms that "girl y" is somewhere to the right of "girl x"
    right(g1, g3).
    right(g2, g4).
    right(g3, g5).
    right(g1, g4).
    right(g2, g5).
    right(g1, g5).
    right(g1, g2).
    right(g4, g5).
    right(g2, g3).
    right(g3, g4).

    -right(g2, g1).
    -right(g3, g2).
    -right(g4, g3).
    -right(g5, g4).
    -right(g1, g1).
    -right(g2, g2).
    -right(g3, g3).
    -right(g4, g4).
    -right(g5, g5).
    -right(g3, g1).
    -right(g4, g2).
    -right(g5, g3).
    -right(g4, g1).
    -right(g5, g2).
    -right(g5, g1).

% Each girl has different characteristics
    Girl1(x) | Girl2(x) | Girl3(x) | Girl4(x) | Girl5(x).
    Ella(x) | Julia(x) | Lily(x) | Makayla(x) | Riley(x).
    Black(x) | Blue(x) | Green(x) | Red(x) | White(x).
    Aztec(x) | Chinese(x) | Egyptian(x) | Greek(x) | Roman(x).
    Architecture(x) | Art(x) | Economy(x) | Religion(x) | Science(x).
    Ten(x) | Eleven(x) | Twelve(x) | Thirteen(x) | Fourteen(x).
    Apple(x) | Cranberry(x) | Grapefruit(x) | Orange(x) | Strawberry(x).

% Each property applies at most to one girl
    Ella(x) & Ella(y) -> -differentFrom(x, y).
    Julia(x) & Julia(y) -> -differentFrom(x, y).
    Lily(x) & Lily(y) -> -differentFrom(x, y).
    Makayla(x) & Makayla(y) -> -differentFrom(x, y).
    Riley(x) & Riley(y) -> -differentFrom(x, y).

    Black(x) & Black(y) -> -differentFrom(x, y).
```

```
206       Blue(x) & Blue(y) -> -differentFrom(x, y).
207       Green(x) & Green(y) -> -differentFrom(x, y).
208       Red(x) & Red(y) -> -differentFrom(x, y).
209       White(x) & White(y) -> -differentFrom(x, y).
210
211       Aztec(x) & Aztec(y) -> -differentFrom(x, y).
212       Chinese(x) & Chinese(y) -> -differentFrom(x, y).
213       Egyptian(x) & Egyptian(y) -> -differentFrom(x, y).
214       Greek(x) & Greek(y) -> -differentFrom(x, y).
215       Roman(x) & Roman(y) -> -differentFrom(x, y).
216
217       Architecture(x) & Architecture(y) -> -differentFrom(x, y).
218       Art(x) & Art(y) -> -differentFrom(x, y).
219       Economy(x) & Economy(y) -> -differentFrom(x, y).
220       Religion(x) & Religion(y) -> -differentFrom(x, y).
221       Science(x) & Science(y) -> -differentFrom(x, y).
222
223       Ten(x) & Ten(y) -> -differentFrom(x, y).
224       Eleven(x) & Eleven(y) -> -differentFrom(x, y).
225       Twelve(x) & Twelve(y) -> -differentFrom(x, y).
226       Thirteen(x) & Thirteen(y) -> -differentFrom(x, y).
227       Fourteen(x) & Fourteen(y) -> -differentFrom(x, y).
228
229       Apple(x) & Apple(y) -> -differentFrom(x, y).
230       Cranberry(x) & Cranberry(y) -> -differentFrom(x, y).
231       Grapefruit(x) & Grapefruit(y) -> -differentFrom(x, y).
232       Orange(x) & Orange(y) -> -differentFrom(x, y).
233       Strawberry(x) & Strawberry(y) -> -differentFrom(x, y).
234
235   %Clues
236
237       % 1. At the fourth position is the girl drinking Strawberry juice.
238       Girl4(x) -> Strawberry(x).
239
240       % 2. The girl drinking Strawberry juice is somewhere between the girl who likes Art
          and the girl that is drinking Apple juice, in that order.
241       Art(x) & Apple(x) -> inbetween(x, y).
242       Strawberry(x) & Art(y) -> right(y, x).
243       Strawberry(x) & Apple(y) -> right(x, y).
244
245       % 3. Riley is next to the girl who likes the Roman civilization.
246       Riley(x) & Roman(y) -> nextto(y, x).
247
248       % 4. The girl wearing the Green shirt is drinking Apple juice.
249       Green(x) -> Apple(x).
250
251       % 5. At the fourth position is the girl that is interested in Religion.
252       Girl4(x) -> Religion(x).
253
254       % 6. The 12 years old girl is exactly to the right of the 11 years old girl.
255       Twelve(x) & Eleven(y) -> directlyright(y, x).
256
257       % 7. At one of the ends is the girl who likes the Egyptian civilization.
258       Egyptian(x) -> extremity(x).
259
260       % 8. The girl wearing the Black shirt is somewhere between the girl that likes
          Science and the girl wearing the Red shirt, in that order.
261       Science(x) & Red(y) -> inbetween(x, y).
262       Black(x) & Science(y) -> right(y, x).
263       Black(x) & Red(y) -> right(x, y).
264
265       % 9. The girl wearing the Red shirt is somewhere to the left of the girl wearing
          the White shirt.
266       Red(x) & White(y) -> right(x, y).
267
268       % 10. The girl drinking Cranberry juice likes the Greek civilization.
269       Cranberry(x) -> Greek(x).
270
271       % 11. Ella is at the third position.
```

```
272       Girl3(x) -> Ella(x).
273
274       % 12. The 12 years old girl is next to the girl that is interested in Architecture.
275       Twelve(x) & Architecture(y) -> nextto(y, x).
276
277       % 13. Riley is exactly to the right of the girl that admires the Roman civilization.
278       Riley(x) & Roman(y) -> directlyright(y, x).
279
280       % 14. The girl drinking Cranberry juice is somewhere between the 13 years old girl
          and the girl drinking Grapefruit juice, in that order.
281       Thirteen(x) & Grapefruit(y) -> inbetween(x, y).
282       Cranberry(x) & Thirteen(y) -> right(y, x).
283       Cranberry(x) & Grapefruit(y) -> right(x, y).
284
285       % 15. At one of the ends is the girl that likes Science.
286       Science(x) -> extremity(x).
287
288       % 16. At one of the ends is the girl who loves Architecture.
289       Architecture(x) -> extremity(x).
290
291       % 17. Lily is exactly to the right of the girl who likes Economy.
292       Lily(x) & Economy(y) -> directlyright(y, x).
293
294       % 18. Makayla is 10.
295       Makayla(x) -> Ten(x).
296
297       % 19. The girl interested in the Aztec civilization is somewhere between the girl
          interested in the Roman civilization and the girl interested in the Chinese
          civilization, in that order
298       Roman(x) & Chinese(y) -> inbetween(x, y).
299       Aztec(x) & Roman(y) -> right(y, x).
300       Aztec(x) & Chinese(y) -> right(x, y).
301
302       % 20. The youngest girl is at one of the ends.
303       Ten(x) -> extremity(x).
304
305       % 21. At the second position is the girl that likes Art.
306       Girl2(x) -> Art(x).
307
308   end_of_list.
309
```

**Travel Agency** (page 1 of 4)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Woman4(d4) -> Italy(d4).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Womans' order in row
    % Woman1: first woman
    % Woman2: second woman
    % Woman3: third woman
    % Woman4: fourth woman
    % Woman5: fifth woman

% Womans' names
    % Ana, Glenda, Jessie, Lara, Rose

% Shirt
    % Blue, Green, Red, White, Yellow

% Age
    % TwentyFour, TwentySix, TwentyEight, Thirty, ThirtyTwo

% Profession
    % Biologist, Hostess, Judge, Nurse, Singer

% Country
    % China, Egypt, Italy, Mexico, Peru

% Duration
    % FiveDays, TenDays, FifteenDays, TwentyDays, TwentyFiveDays

formulas(assumptions).

% The 5 womans differ
    differentFrom(d1, d2).
    differentFrom(d1, d3).
    differentFrom(d1, d4).
    differentFrom(d1, d5).
    differentFrom(d2, d3).
    differentFrom(d2, d4).
    differentFrom(d2, d5).
    differentFrom(d3, d4).
    differentFrom(d3, d5).
    differentFrom(d4, d5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    d1 = 0.
    d2 = 1.
    d3 = 2.
    d4 = 3.
    d5 = 4.

    Woman1(0).
    Woman2(1).
    Woman3(2).
    Woman4(3).
    Woman5(4).

% Relation directlyright(x, y) that affirms that "woman y" is directly to the right of
```

**Travel Agency** (page 2 of 4)

```
    "woman x"
70      directlyright(d1, d2).
71      directlyright(d2, d3).
72      directlyright(d3, d4).
73      directlyright(d4, d5).
74
75      -directlyright(d1, d1).
76      -directlyright(d2, d2).
77      -directlyright(d3, d3).
78      -directlyright(d4, d4).
79      -directlyright(d5, d5).
80      -directlyright(d1, d3).
81      -directlyright(d2, d4).
82      -directlyright(d3, d5).
83      -directlyright(d3, d1).
84      -directlyright(d4, d2).
85      -directlyright(d5, d3).
86      -directlyright(d2, d1).
87      -directlyright(d3, d2).
88      -directlyright(d4, d3).
89      -directlyright(d5, d4).
90      -directlyright(d1, d4).
91      -directlyright(d2, d5).
92      -directlyright(d4, d1).
93      -directlyright(d5, d2).
94      -directlyright(d1, d5).
95      -directlyright(d5, d1).
96
97  % Relation nextto(x, y) that affirms that "woman y" is next to "woman x"
98      directlyright(x, y) | directlyright(y, x) <-> nextto(x, y).
99
100 % Relation extremity(x) that affirms that "woman x" is at One of the ends.
101     extremity(d1).
102     extremity(d5).
103
104     -extremity(d2).
105     -extremity(d3).
106     -extremity(d4).
107
108 % Relation right(x, y) that affirms that "woman y" is somewhere to the right of "woman x"
109     right(d1, d3).
110     right(d2, d4).
111     right(d3, d5).
112     right(d1, d4).
113     right(d2, d5).
114     right(d1, d5).
115     right(d1, d2).
116     right(d4, d5).
117     right(d2, d3).
118     right(d3, d4).
119
120     -right(d2, d1).
121     -right(d3, d2).
122     -right(d4, d3).
123     -right(d5, d4).
124     -right(d1, d1).
125     -right(d2, d2).
126     -right(d3, d3).
127     -right(d4, d4).
128     -right(d5, d5).
129     -right(d3, d1).
130     -right(d4, d2).
131     -right(d5, d3).
132     -right(d4, d1).
133     -right(d5, d2).
134     -right(d5, d1).
135
136 % Each woman has different characteristics
137     Woman1(x) | Woman2(x) | Woman3(x) | Woman4(x) | Woman5(x).
```

```
138       Ana(x) | Glenda(x) | Jessie(x) | Lara(x) | Rose(x).
139       Blue(x) | Green(x) | Red(x) | White(x) | Yellow(x).
140       TwentyFour(x) | TwentySix(x) | TwentyEight(x) | Thirty(x) | ThirtyTwo(x).
141       Biologist(x) | Hostess(x) | Judge(x) | Nurse(x) | Singer(x).
142       China(x) | Egypt(x) | Italy(x) | Mexico(x) | Peru(x).
143       FiveDays(x) | TenDays(x) | FifteenDays(x) | TwentyDays(x) | TwentyFiveDays(x).
144
145   % Each property applies at most to one woman
146       Ana(x) & Ana(y) -> -differentFrom(x, y).
147       Glenda(x) & Glenda(y) -> -differentFrom(x, y).
148       Jessie(x) & Jessie(y) -> -differentFrom(x, y).
149       Lara(x) & Lara(y) -> -differentFrom(x, y).
150       Rose(x) & Rose(y) -> -differentFrom(x, y).
151
152       Blue(x) & Blue(y) -> -differentFrom(x, y).
153       Green(x) & Green(y) -> -differentFrom(x, y).
154       Red(x) & Red(y) -> -differentFrom(x, y).
155       White(x) & White(y) -> -differentFrom(x, y).
156       Yellow(x) & Yellow(y) -> -differentFrom(x, y).
157
158       TwentyFour(x) & TwentyFour(y) -> -differentFrom(x, y).
159       TwentySix(x) & TwentySix(y) -> -differentFrom(x, y).
160       TwentyEight(x) & TwentyEight(y) -> -differentFrom(x, y).
161       Thirty(x) & Thirty(y) -> -differentFrom(x, y).
162       ThirtyTwo(x) & ThirtyTwo(y) -> -differentFrom(x, y).
163
164       Biologist(x) & Biologist(y) -> -differentFrom(x, y).
165       Hostess(x) & Hostess(y) -> -differentFrom(x, y).
166       Judge(x) & Judge(y) -> -differentFrom(x, y).
167       Nurse(x) & Nurse(y) -> -differentFrom(x, y).
168       Singer(x) & Singer(y) -> -differentFrom(x, y).
169
170       China(x) & China(y) -> -differentFrom(x, y).
171       Egypt(x) & Egypt(y) -> -differentFrom(x, y).
172       Italy(x) & Italy(y) -> -differentFrom(x, y).
173       Mexico(x) & Mexico(y) -> -differentFrom(x, y).
174       Peru(x) & Peru(y) -> -differentFrom(x, y).
175
176       FiveDays(x) & FiveDays(y) -> -differentFrom(x, y).
177       TenDays(x) & TenDays(y) -> -differentFrom(x, y).
178       FifteenDays(x) & FifteenDays(y) -> -differentFrom(x, y).
179       TwentyDays(x) & TwentyDays(y) -> -differentFrom(x, y).
180       TwentyFiveDays(x) & TwentyFiveDays(y) -> -differentFrom(x, y).
181
182   %Clues
183
184       % 1. The Singer is at the third position.
185       Woman3(x) -> Singer(x).
186
187       % 2. The woman traveling for 20 days is somewhere between the woman who is going to
          Peru and the owner of the Blue purse, in that order.
188       Peru(x) & TwentyDays(y) & Blue(z) -> right(x, y) & right(y, z).
189
190       % 3. Ana is exactly to the left of the Biologist.
191       Ana(x) & Biologist(y) -> directlyright(x, y).
192
193       % 4. The 32 years old is going to see the Sahara.
194       ThirtyTwo(x) -> Egypt(x).
195
196       % 5. The owner of the White purse is exactly to the right of the woman traveling to
          visit Machu Picchu.
197       Peru(x) & White(y) -> directlyright(x, y).
198
199       % 6. Glenda is somewhere to the right of the woman who has the Green purse.
200       Green(x) & Glenda(y) -> right(x, y).
201
202       % 7. The person wearing the White purse is somewhere between the 30 years old woman
          and the owner of the Blue purse, in that order.
203       Thirty(x) & White(y) & Blue(z) -> right(x, y) & right(y, z).
```

```
204
205       % 8. The 24 years old woman is going to visit an Aztec pyramid.
206       TwentyFour(x) -> Mexico(x).
207
208       % 9. The woman wearing the White purse is somewhere to the left of the youngest
          woman.
209       White(x) & TwentyFour(y) -> right(x, y).
210
211       % 10. The traveler going to Italy is exactly to the right of the woman traveling
          for 20 days.
212       TwentyDays(x) & Italy(y) -> directlyright(x, y).
213
214       % 11. The person who is going to travel for 25 days has the Red purse.
215       TwentyFiveDays(x) -> Red(x).
216
217       % 12. The Judge is in the first position.
218       Woman1(x) -> Judge(x).
219
220       % 13. The Nurse is exactly to the right of the woman who is going to travel for 20
          days.
221       TwentyDays(x) & Nurse(y) -> directlyright(x, y).
222
223       % 14. The Hostess is somewhere between Lara and the woman who has the Blue purse,
          in that order.
224       Lara(x) & Hostess(y) & Blue(z) -> right(x, y) & right(y, z).
225
226       % 15. In the second position is the woman that is going to travel for 15 days.
227       Woman2(x) -> FifteenDays(x).
228
229       % 16. Rose has the Green purse.
230       Rose(x) -> Green(x).
231
232       % 17. The woman who is traveling for less than a week is exactly to the left of the
          32 years old woman.
233       FiveDays(x) & ThirtyTwo(y) -> directlyright(x, y).
234
235       % 18. The person traveling for 5 days is 28.
236       FiveDays(x) -> TwentyEight(x).
237
238       % 19. The Blue purse owner is somewhere between the 30 years old woman and the
          owner of the Yellow purse, in that order.
239       Thirty(x) & Blue(y) & Yellow(z) -> right(x, y) & right(y, z).
240
241   end_of_list.
242
```

**Persian Rugs** (page 1 of 4)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Man4(d4) -> Round(d4).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Men's order in row
    % Man1: first man
    % Man2: second man
    % Man3: third man
    % Man4: fourth man
    % Man5: fifth man

% Men's names
    % Bryan, Casey, Garrett, Shane, Timothy

% Colour
    % Blue, Green, Orange, Red, Yellow

% Type
    % Oval, Rectangular, Round, Runner, Square

% Symbols
    % Birds, Diamonds, Lotuses, Roses, Stars

% Price
    % FourHundred, SixHundred, EightHundred, OneThousand, TwelveHundred

% Place
    % Bedroom, Hall, Library, LivingRoom, Office

formulas(assumptions).

% The 5 men differ
    differentFrom(d1, d2).
    differentFrom(d1, d3).
    differentFrom(d1, d4).
    differentFrom(d1, d5).
    differentFrom(d2, d3).
    differentFrom(d2, d4).
    differentFrom(d2, d5).
    differentFrom(d3, d4).
    differentFrom(d3, d5).
    differentFrom(d4, d5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    d1 = 0.
    d2 = 1.
    d3 = 2.
    d4 = 3.
    d5 = 4.

    Man1(0).
    Man2(1).
    Man3(2).
    Man4(3).
    Man5(4).

% Relation directlyright(x, y) that affirms that "man y" is directly to the right of
```

```
    "man x"
70      directlyright(d1, d2).
71      directlyright(d2, d3).
72      directlyright(d3, d4).
73      directlyright(d4, d5).
74
75      -directlyright(d1, d1).
76      -directlyright(d2, d2).
77      -directlyright(d3, d3).
78      -directlyright(d4, d4).
79      -directlyright(d5, d5).
80      -directlyright(d1, d3).
81      -directlyright(d2, d4).
82      -directlyright(d3, d5).
83      -directlyright(d3, d1).
84      -directlyright(d4, d2).
85      -directlyright(d5, d3).
86      -directlyright(d2, d1).
87      -directlyright(d3, d2).
88      -directlyright(d4, d3).
89      -directlyright(d5, d4).
90      -directlyright(d1, d4).
91      -directlyright(d2, d5).
92      -directlyright(d4, d1).
93      -directlyright(d5, d2).
94      -directlyright(d1, d5).
95      -directlyright(d5, d1).
96
97  % Relation nextto(x, y) that affirms that "man y" is next to "man x"
98      directlyright(x, y) | directlyright(y, x) <-> nextto(x, y).
99
100 % Relation extremity(x) that affirms that "man x" is at One of the ends.
101     extremity(d1).
102     extremity(d5).
103
104     -extremity(d2).
105     -extremity(d3).
106     -extremity(d4).
107
108 % Relation right(x, y) that affirms that "man y" is somewhere to the right of "man x"
109     right(d1, d3).
110     right(d2, d4).
111     right(d3, d5).
112     right(d1, d4).
113     right(d2, d5).
114     right(d1, d5).
115     right(d1, d2).
116     right(d4, d5).
117     right(d2, d3).
118     right(d3, d4).
119
120     -right(d2, d1).
121     -right(d3, d2).
122     -right(d4, d3).
123     -right(d5, d4).
124     -right(d1, d1).
125     -right(d2, d2).
126     -right(d3, d3).
127     -right(d4, d4).
128     -right(d5, d5).
129     -right(d3, d1).
130     -right(d4, d2).
131     -right(d5, d3).
132     -right(d4, d1).
133     -right(d5, d2).
134     -right(d5, d1).
135
136 % % Each man has different characteristics
137     Man1(x) | Man2(x) | Man3(x) | Man4(x) | Man5(x).
```

```
138       Bryan(x) | Casey(x) | Garrett(x) | Shane(x) | Timothy(x).
139       Blue(x) | Green(x) | Orange(x) | Red(x) | Yellow(x).
140       Oval(x) | Rectangular(x) | Round(x) | Runner(x) | Square(x).
141       Birds(x) | Diamonds(x) | Lotuses(x) | Roses(x) | Stars(x).
142       FourHundred(x) | SixHundred(x) | EightHundred(x) | OneThousand(x) | TwelveHundred(x).
143       Bedroom(x) | Hall(x) | Library(x) | LivingRoom(x) | Office(x).
144
145   % Each property applies at most to one man
146       Bryan(x) & Bryan(y) -> -differentFrom(x, y).
147       Casey(x) & Casey(y) -> -differentFrom(x, y).
148       Garrett(x) & Garrett(y) -> -differentFrom(x, y).
149       Shane(x) & Shane(y) -> -differentFrom(x, y).
150       Timothy(x) & Timothy(y) -> -differentFrom(x, y).
151
152       Blue(x) & Blue(y) -> -differentFrom(x, y).
153       Green(x) & Green(y) -> -differentFrom(x, y).
154       Orange(x) & Orange(y) -> -differentFrom(x, y).
155       Red(x) & Red(y) -> -differentFrom(x, y).
156       Yellow(x) & Yellow(y) -> -differentFrom(x, y).
157
158       Oval(x) & Oval(y) -> -differentFrom(x, y).
159       Rectangular(x) & Rectangular(y) -> -differentFrom(x, y).
160       Round(x) & Round(y) -> -differentFrom(x, y).
161       Runner(x) & Runner(y) -> -differentFrom(x, y).
162       Square(x) & Square(y) -> -differentFrom(x, y).
163
164       Birds(x) & Birds(y) -> -differentFrom(x, y).
165       Diamonds(x) & Diamonds(y) -> -differentFrom(x, y).
166       Lotuses(x) & Lotuses(y) -> -differentFrom(x, y).
167       Roses(x) & Roses(y) -> -differentFrom(x, y).
168       Stars(x) & Stars(y) -> -differentFrom(x, y).
169
170       FourHundred(x) & FourHundred(y) -> -differentFrom(x, y).
171       SixHundred(x) & SixHundred(y) -> -differentFrom(x, y).
172       EightHundred(x) & EightHundred(y) -> -differentFrom(x, y).
173       OneThousand(x) & OneThousand(y) -> -differentFrom(x, y).
174       TwelveHundred(x) & TwelveHundred(y) -> -differentFrom(x, y).
175
176       Bedroom(x) & Bedroom(y) -> -differentFrom(x, y).
177       Hall(x) & Hall(y) -> -differentFrom(x, y).
178       Library(x) & Library(y) -> -differentFrom(x, y).
179       LivingRoom(x) & LivingRoom(y) -> -differentFrom(x, y).
180       Office(x) & Office(y) -> -differentFrom(x, y).
181
182   %Clues
183
184       % 1. The man who bought the $400 rug is exactly to the left of the man who bought
          the rug with Birds.
185       FourHundred(x) & Birds(y) -> directlyright(x, y).
186
187       % 2. Bryan is next to the client that got the Runner rug.
188       Bryan(x) & Runner(y) -> nextto(y, x).
189
190       % 3. The Red rug's owner is somewhere to the left of the man who bought a rug for
          his Hall.
191       Red(x) & Hall(y) -> right(x, y).
192
193       % 4. The buyer who got the most expensive rug is next to the buyer who got the
          Round rug.
194       TwelveHundred(x) & Round(y) -> nextto(y, x).
195
196       % 5. The Yellow rug's owner is somewhere between the Oval rug's owner and the man
          that spent $800, in that order.
197       Oval(x) & Yellow(y) & EightHundred(z) -> right(x, y) & right(y, z).
198
199       % 6. The man that bought the Red rug is exactly to the left of the man that bought
          a rug for his Office.
200       Red(x) & Office(y) -> directlyright(x, y).
201
```

```
202       % 7. Garrett is next to the customer who spent $600.
203       Garrett(x) & SixHundred(y) -> nextto(y, x).
204
205       % 8. Shane is somewhere to the right of the Orange rug's owner.
206       Shane(x) & Orange(y) -> right(y, x).
207
208       % 9. The cheapest rug has Stars on it.
209       FourHundred(x) -> Stars(x).
210
211       % 10. At the fifth position is the client who bought the Rectangular rug.
212       Man5(x) -> Rectangular(x).
213
214       % 11. The man who bought the $600 rug is exactly to the left of the man who bought
          the rug with Roses.
215       SixHundred(x) & Roses(y) -> directlyright(x, y).
216
217       % 12. Casey is somewhere to the right of the customer that got the Yellow rug.
218       Casey(x) & Yellow(y) -> right(y, x).
219
220       % 13. The buyer who purchased the Round rug is exactly to the left of the man who
          spent $800.
221       Round(x) & EightHundred(y) -> directlyright(x, y).
222
223       % 14. The man that bought the cheapest rug is exactly to the left of the man that
          acquired the Square rug.
224       FourHundred(x) & Square(y) -> directlyright(x, y).
225
226       % 15. At the third position is the man that got a rug for his Bedroom.
227       Man3(x) -> Bedroom(x).
228
229       % 16. The customer that bought the rug with Stars is exactly to the right of the
          customer that bought the Green rug.
230       Stars(x) & Green(y) -> directlyright(y, x).
231
232       % 17. Shane is next to the man who acquired the Round rug.
233       Shane(x) & Round(y) -> nextto(y, x).
234
235       % 18. The man who got the rug with Birds is somewhere between the man who got the
          Yellow rug and the man who got the rug with Lotuses, in that order.
236       Yellow(x) & Birds(y) & Lotuses(z) -> right(x, y) & right(y, z).
237
238       % 19. The customer that spent $800 is somewhere to the right of the Red rug's owner.
239       EightHundred(x) & Red(y) -> right(y, x).
240
241       % 20. The client that bought a rug for his Living room is exactly to the right of
          the client that bought a rug for his Library.
242       LivingRoom(x) & Library(y) -> directlyright(y, x).
243
244   end_of_list.
245
```

**Secret Agents** (page 1 of 4)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Agent4(d4) -> James(d4).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Agents' order in row
    % Agent1: first agent
    % Agent2: second agent
    % Agent3: third agent
    % Agent4: fourth agent
    % Agent5: fifth agent

% Agents' names
    % Austin, James, Jason, Stan, Sterling

% Tie
    % Black, Blue, Green, Purple, Red

% Country
    % Australia, Brazil, Germany, Libya, Russia

% Accessory
    % Clock, Pen, Phone, Ring, Umbrella

% Skill
    % ComputerHacking, Disguise, Driving, MartialArts, Parkour

% Age
    % TwentyFive, Thirty, ThirtyFive, Forty, FortyFive

formulas(assumptions).

% The 5 agents differ
    differentFrom(d1, d2).
    differentFrom(d1, d3).
    differentFrom(d1, d4).
    differentFrom(d1, d5).
    differentFrom(d2, d3).
    differentFrom(d2, d4).
    differentFrom(d2, d5).
    differentFrom(d3, d4).
    differentFrom(d3, d5).
    differentFrom(d4, d5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    d1 = 0.
    d2 = 1.
    d3 = 2.
    d4 = 3.
    d5 = 4.

    Agent1(0).
    Agent2(1).
    Agent3(2).
    Agent4(3).
    Agent5(4).

% Relation directlyright(x, y) that affirms that "agent y" is directly to the right of
```

```
    "agent x".
70      directlyright(d1, d2).
71      directlyright(d2, d3).
72      directlyright(d3, d4).
73      directlyright(d4, d5).
74
75      -directlyright(d1, d1).
76      -directlyright(d2, d2).
77      -directlyright(d3, d3).
78      -directlyright(d4, d4).
79      -directlyright(d5, d5).
80      -directlyright(d1, d3).
81      -directlyright(d2, d4).
82      -directlyright(d3, d5).
83      -directlyright(d3, d1).
84      -directlyright(d4, d2).
85      -directlyright(d5, d3).
86      -directlyright(d2, d1).
87      -directlyright(d3, d2).
88      -directlyright(d4, d3).
89      -directlyright(d5, d4).
90      -directlyright(d1, d4).
91      -directlyright(d2, d5).
92      -directlyright(d4, d1).
93      -directlyright(d5, d2).
94      -directlyright(d1, d5).
95      -directlyright(d5, d1).
96
97  % Relation nextto(x, y) that affirms that "agent y" is next to "agent x".
98      directlyright(x, y) | directlyright(y, x) <-> nextto(x, y).
99
100 % Relation extremity(x) that affirms that "agent x" is at One of the ends.
101     extremity(d1).
102     extremity(d5).
103
104     -extremity(d2).
105     -extremity(d3).
106     -extremity(d4).
107
108 % Relation right(x, y) that affirms that "agent y" is somewhere to the right of "agent
    x".
109     right(d1, d3).
110     right(d2, d4).
111     right(d3, d5).
112     right(d1, d4).
113     right(d2, d5).
114     right(d1, d5).
115     right(d1, d2).
116     right(d4, d5).
117     right(d2, d3).
118     right(d3, d4).
119
120     -right(d2, d1).
121     -right(d3, d2).
122     -right(d4, d3).
123     -right(d5, d4).
124     -right(d1, d1).
125     -right(d2, d2).
126     -right(d3, d3).
127     -right(d4, d4).
128     -right(d5, d5).
129     -right(d3, d1).
130     -right(d4, d2).
131     -right(d5, d3).
132     -right(d4, d1).
133     -right(d5, d2).
134     -right(d5, d1).
135
136 % Each agent has different characteristics
```

```
137       Agent1(x) | Agent2(x) | Agent3(x) | Agent4(x) | Agent5(x).
138       Austin(x) | James(x) | Jason(x) | Stan(x) | Sterling(x).
139       Black(x) | Blue(x) | Green(x) | Purple(x) | Red(x).
140       Australia(x) | Brazil(x) | Germany(x) | Libya(x) | Russia(x).
141       Clock(x) | Pen(x) | Phone(x) | Ring(x) | Umbrella(x).
142       ComputerHacking(x) | Disguise(x) | Driving(x) | MartialArts(x) | Parkour(x).
143       TwentyFive(x) | Thirty(x) | ThirtyFive(x) | Forty(x) | FortyFive(x).
144
145   % Each property applies at most to one agent
146       Austin(x) & Austin(y) -> -differentFrom(x, y).
147       James(x) & James(y) -> -differentFrom(x, y).
148       Jason(x) & Jason(y) -> -differentFrom(x, y).
149       Stan(x) & Stan(y) -> -differentFrom(x, y).
150       Sterling(x) & Sterling(y) -> -differentFrom(x, y).
151
152       Black(x) & Black(y) -> -differentFrom(x, y).
153       Blue(x) & Blue(y) -> -differentFrom(x, y).
154       Green(x) & Green(y) -> -differentFrom(x, y).
155       Purple(x) & Purple(y) -> -differentFrom(x, y).
156       Red(x) & Red(y) -> -differentFrom(x, y).
157
158       Australia(x) & Australia(y) -> -differentFrom(x, y).
159       Brazil(x) & Brazil(y) -> -differentFrom(x, y).
160       Germany(x) & Germany(y) -> -differentFrom(x, y).
161       Libya(x) & Libya(y) -> -differentFrom(x, y).
162       Russia(x) & Russia(y) -> -differentFrom(x, y).
163
164       Clock(x) & Clock(y) -> -differentFrom(x, y).
165       Pen(x) & Pen(y) -> -differentFrom(x, y).
166       Phone(x) & Phone(y) -> -differentFrom(x, y).
167       Ring(x) & Ring(y) -> -differentFrom(x, y).
168       Umbrella(x) & Umbrella(y) -> -differentFrom(x, y).
169
170       ComputerHacking(x) & ComputerHacking(y) -> -differentFrom(x, y).
171       Disguise(x) & Disguise(y) -> -differentFrom(x, y).
172       Driving(x) & Driving(y) -> -differentFrom(x, y).
173       MartialArts(x) & MartialArts(y) -> -differentFrom(x, y).
174       Parkour(x) & Parkour(y) -> -differentFrom(x, y).
175
176       TwentyFive(x) & TwentyFive(y) -> -differentFrom(x, y).
177       Thirty(x) & Thirty(y) -> -differentFrom(x, y).
178       ThirtyFive(x) & ThirtyFive(y) -> -differentFrom(x, y).
179       Forty(x) & Forty(y) -> -differentFrom(x, y).
180       FortyFive(x) & FortyFive(y) -> -differentFrom(x, y).
181
182   %Clues
183
184       %1. Austin is next to the agent wearing the Black tie.
185       Austin(x) & Black(y) -> nextto(y, x).
186
187       %2. The master of Disguise is exactly to the right of the agent that has a spy
          Umbrella.
188       Disguise(x) & Umbrella(y) -> directlyright(y, x).
189
190       %3. The 35 years old agent is going to a mission on Tripoli.
191       ThirtyFive(x) -> Libya(x).
192
193       %4. James is the youngest secret agent.
194       James(x) -> TwentyFive(x).
195
196       %5. The agent that is going to Australia is next to the agent that is specialized
          in Parkour.
197       Australia(x) & Parkour(y) -> nextto(y, x).
198
199       %6. James is exactly to the right of the agent that has a special Clock.
200       James(x) & Clock(y) -> directlyright(y, x).
201
202       %7. The spy that has an unique Umbrella is between the 40 years old agent and
          Austin, in that order.
```

```
203       Forty(x) & Umbrella(y) & Austin(z) -> right(x, y) & right(y, z).
204
205       %8. Stan is next to the agent that is going to Asia.
206       Stan(x) & Russia(y) -> nextto(y, x).
207
208       %9. Sterling is at one of the ends.
209       Sterling(x) -> extremity(x).
210
211       %10. The man wearing the Red tie is 40 years old.
212       Red(x) -> Forty(x).
213
214       %11. The spy that is going to South America is exactly to the left of the 45 years
          old spy.
215       Brazil(x) & FortyFive(y) -> directlyright(x, y).
216
217       %12. Jason is exactly to the left of Austin.
218       Jason(x) & Austin(y) -> directlyright(x, y).
219
220       %13. The Driver expert is next to the 30 years old man.
221       Driving(x) & Thirty(y) -> nextto(y, x).
222
223       %14. The 35 years old agent is next to the agent that is going to Sydney.
224       ThirtyFive(x) & Australia(y) -> nextto(y, x).
225
226       %15. The agent with advanced knowledge of Hacking is exactly to the left of the 35
          years old man.
227       ComputerHacking(x) & ThirtyFive(y) -> directlyright(x, y).
228
229       %16. The spy wearing the Purple tie is next to the Geek spy.
230       Purple(x) & ComputerHacking(y) -> nextto(y, x).
231
232       %17. Austin is 30.
233       Austin(x) -> Thirty(x).
234
235       %18. The agent that has a special Phone is exactly to the left of the agent that is
          going to Africa.
236       Phone(x) & Libya(y) -> directlyright(x, y).
237
238       %19. The agent wearing a spy Ring is somewhere to the right of the agent wearing
          the Purple tie.
239       Ring(x) & Purple(y) -> right(y, x).
240
241       %20. At the second position is the spy wearing the Green tie.
242       Agent2(x) -> Green(x).
243
244       %21. The spy that is going to Australia is exactly to the right of the 30 years old
          spy.
245       Australia(x) & Thirty(y) -> directlyright(y, x).
246
247   end_of_list.
248
```

### A.1.3 Planning

**Packages with bombs - Domain** (page 1 of 2)

```
(define (domain btnd)
    (:types package bomb toilet)
    (:predicates
        (in ?p - package ?b - bomb)
        (defused ?b - bomb)
        (clog ?toilet - toilet)
        (stuck ?toilet - toilet))

    (:action senseIN
        :parameters (?p - package ?b - bomb)
        :observe (in ?p ?b))

    (:action dunk
        :parameters (?p - package ?b - bomb ?t - toilet)
        :precondition (and (not (clog ?t)) (not (stuck ?t)))
        :effect (and (when (in ?p ?b) (defused ?b))
            (clog ?t)
            (nondet (stuck ?t))))

    (:action flush
        :parameters (?t - toilet)
        :effect (not (clog ?t)))

    (:action unstick
        :parameters (?toilet - toilet)
        :effect (and (when (stuck ?toilet) (not (stuck ?toilet))))))
```

**Packages with bombs - Problem** (page 2 of 2)

```
(define (problem btnd - 4)
    (:domain btnd)
    (:objects
        b0 b1 - bomb
        p0 p1 p2 p3 - package
        t0 - toilet)

    (:init
        (in p0 b0)
        (unknown (in p1 b1))
        (unknown (in p2 b1))
        (unknown (in p3 b1))
        (oneof (in p1 b1) (in p2 b1) (in p3 b1)))
    (:goal (and (defused b0) (defused b1))))

```

**Sliding Tiles - Domain** (page 1 of 1)

```
(define (domain sliding-tile)
    (:predicates
        (tile-at ?t ?r ?c)
        (is-blank ?r ?c)
        (next-row ?r1 ?r2)
        (next-col ?c1 ?c2))
    (:action move-tile-down
        :parameters (?tile ?old-row ?new-row ?col)
        :precondition (and (next-row ?old-row ?new-row)
                    (tile-at ?tile ?old-row ?col)
                    (is-blank ?new-row ?col))
        :effect         (and (not (tile-at ?tile ?old-row ?col))
                     (not (is-blank ?new-row ?col))
                     (tile-at ?tile ?new-row ?col)
                     (is-blank ?old-row ?col)))
    (:action move-tile-up
        :parameters(?tile ?old-row ?new-row ?col)
        :precondition (and (next-row ?new-row ?old-row)
                    (tile-at ?tile ?old-row ?col)
                     (is-blank ?new-row ?col))
        :effect(and (not (tile-at ?tile ?old-row ?col))
                 (not (is-blank ?new-row ?col))
                 (tile-at ?tile ?new-row ?col)
                 (is-blank ?old-row ?col)))
    (:action move-tile-right
        :parameters(?tile ?row ?old-col ?new-col)
        :precondition (and (next-col ?old-col ?new-col)
                    (tile-at ?tile ?row ?old-col)
                     (is-blank ?row ?new-col))
        :effect(and (not (tile-at ?tile ?row ?old-col))
                 (not (is-blank ?row ?new-col))
                 (tile-at ?tile ?row ?new-col)
                 (is-blank ?row ?old-col)))
(:action move-tile-left
        :parameters(?tile ?row ?old-col ?new-col)
        :precondition (and (next-col ?new-col ?old-col)
                    (tile-at ?tile ?row ?old-col)
                     (is-blank ?row ?new-col))
        :effect(and (not (tile-at ?tile ?row ?old-col))
                 (not (is-blank ?row ?new-col))
                 (tile-at ?tile ?row ?new-col)
                 (is-blank ?row ?old-col)))
                 )

```

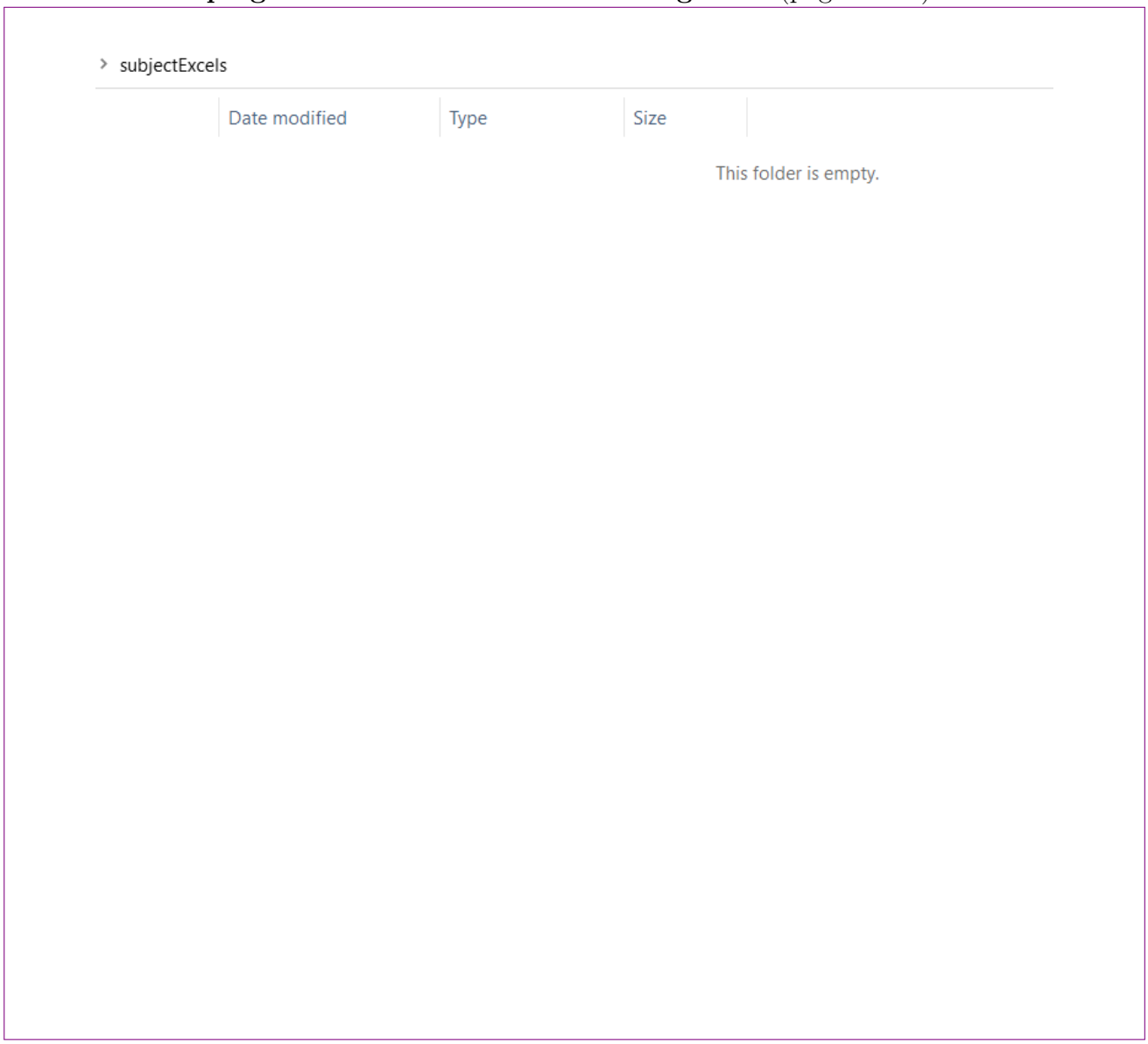
**BONUS - Scraping Bot - Code** (page 1 of 8)

```
from selenium import webdriver

import time
import xlsxwriter

PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)

NUTRIPROFITS_OFFERS_LINK = "https://nutriprofits.com/account/offers"
EXEL_NAME_START = "./"
EXEL_NAME_END = ".xlsx"

SUBJECT_EXCEL = "subjectExcels/"

def login ():
    loginLabel =
    driver.find_element_by_xpath("/html/body/div/div/div/div/div[2]/form/div[1]/input")
    loginLabel.clear()
    loginLabel.send_keys("stoianiuliatia@gmail.com")

    passwordLabel =
    driver.find_element_by_xpath("/html/body/div/div/div/div/div[2]/form/div[2]/input")
    passwordLabel.clear()
    passwordLabel.send_keys("scrapingbot")

    loginButton =
    driver.find_element_by_xpath("/html/body/div/div/div/div/div[2]/form/div[3]/div[2]/bu
    tton")
    loginButton.click()

def createExcelWithProducts (subjectName, listOfProducts): # cream un nou excel pt
fiecare subiect
    # si introducem in ele lista produusele specifice acelui subiect
    excelName = SUBJECT_EXCEL + subjectName + EXEL_NAME_END
    workbook = xlsxwriter.Workbook(excelName)
    worksheet = workbook.add_worksheet()
    data = ([product, 0] for product in listOfProducts)

    rowCount = 0
    for dataProduct, noUse in (data):
        worksheet.write(rowCount, 0, dataProduct)
        rowCount += 1

    workbook.close()

def populatingExcelsWithSubjects(rows, subjectNamesSet):
    for subjectName in subjectNamesSet:
        currentListOfProducts = []
        for row in rows:
            columns = row.find_elements_by_tag_name("td")
            if len(columns) > 0:
                currentProductName = str(columns[1].text) # luam numele produsului
                currentSubjectName = str(columns[2].text) # categoria din care face
                parte produsul
                if currentSubjectName == subjectName:
                    currentListOfProducts.append(currentProductName)
        # now we have CURRENTLISTOFPRODUCTS products of SUBJECTNAME subject
        # create an excel with that information
        createExcelWithProducts(subjectName, currentListOfProducts)


driver.get(NUTRIPROFITS_OFFERS_LINK)
time.sleep(5) # pana se incarca , ca sa putem facem loginu

login()
time.sleep(5) # asteptam pana se incarca dupa ce face loginu

```

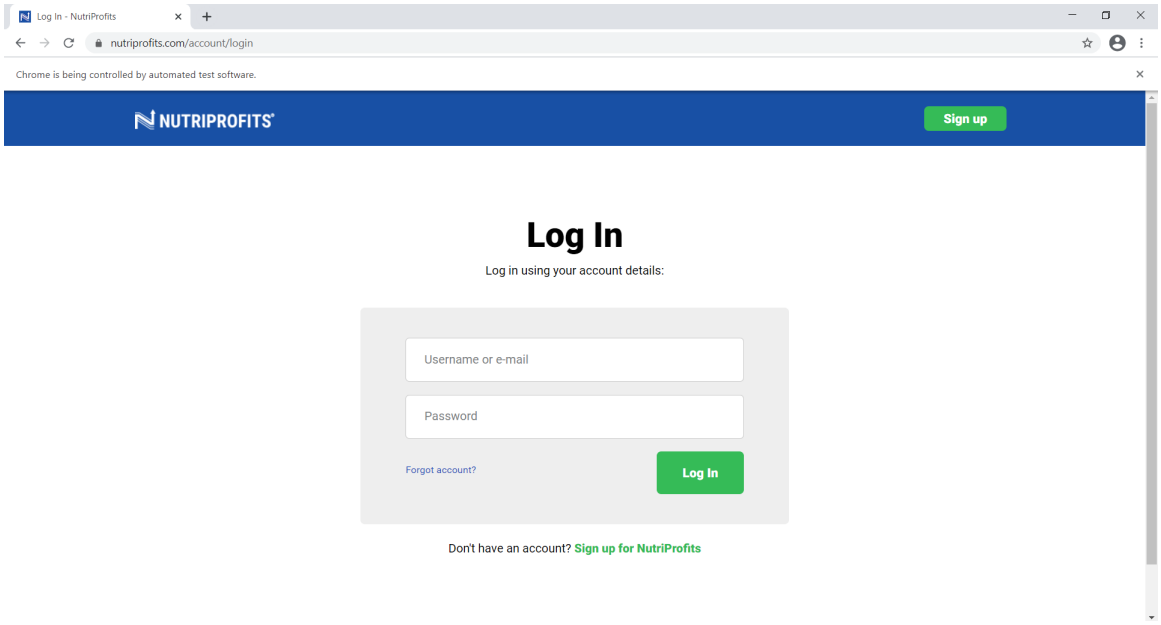
**BONUS - Scraping Bot - Code** (page 2 of 8)

```
62  offersButton = driver.find_element_by_xpath("/html/body/nav[2]/div/div/ul/li[2]/a")
63  offersButton.click()
64  time.sleep(5)
65
66  tableData = driver.find_element_by_xpath("/html/body/div[2]/div/table")
67  rows = tableData.find_elements_by_tag_name("tr") # get all of the rows
68
69
70  subjectNamesSet = set() # iar aici salvam toate categoriile de pe site
71  for row in rows: # primu rand e primu al tabelului => nu e data pe el
72      # get the columns that you want
73      columns = row.find_elements_by_tag_name("td")
74      if len(columns) > 0:
75          currentSubjectName = str(columns[2].text) # retinem doar subiectul
76          subjectNamesSet.add(currentSubjectName) # pentru a crea setul din care vom crea
            excelulurile
77
78  populatingExcelsWithSubjects(rows, subjectNamesSet)
79
80  # BOOM. acum avem in folderu subjectExcels al acestui proiect, cate un excel cu toate
    subiectele.
81  # in fiecare din acestea am introdus produsele specifice
82
83  print "am terminat"
84
85  # in care vom pastra produsele din categoria respectiva
86  # si mailurile oamenilor interesati de categoria respectiva de produse
87
88  """
89  """
90  time.sleep(60) # delay ca sa putemm vedea ce face asta
91  driver.quit()
92
```

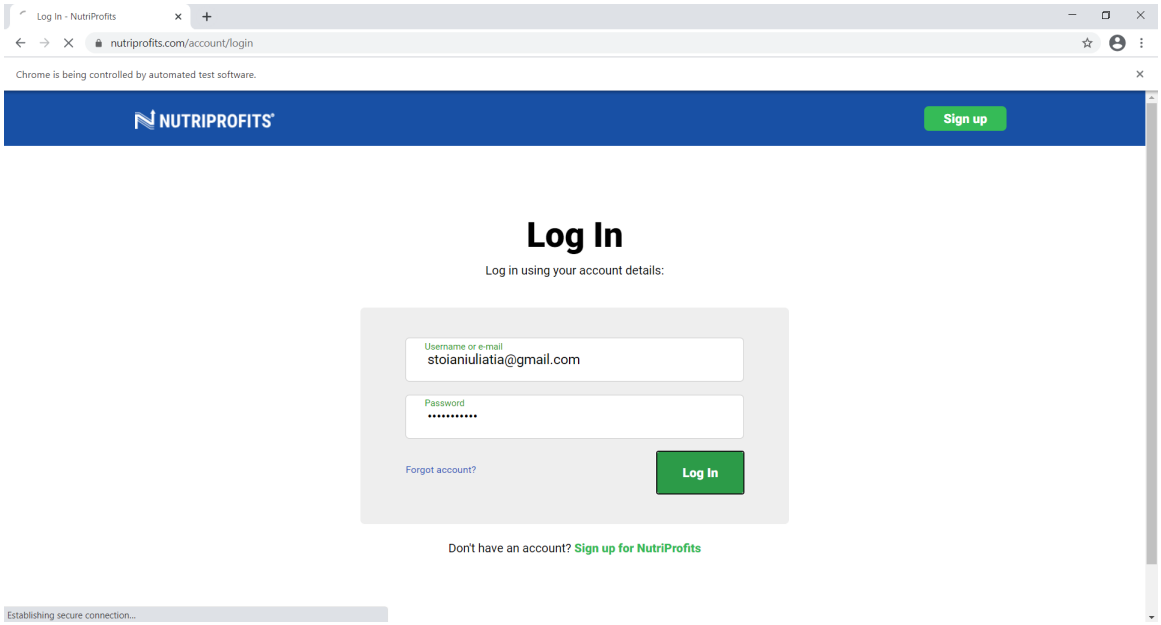
**BONUS - Scraping Bot - Folder Before Collecting Data** (page 3 of 8)



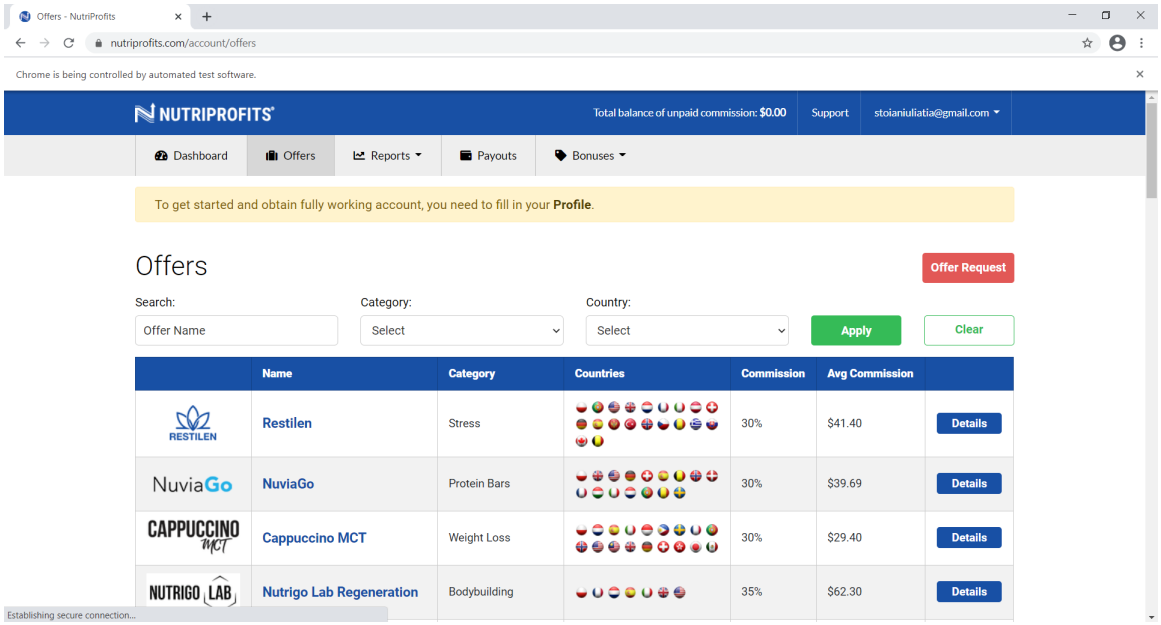
## **BONUS - Scraping Bot - Before Login** (page 4 of 8)



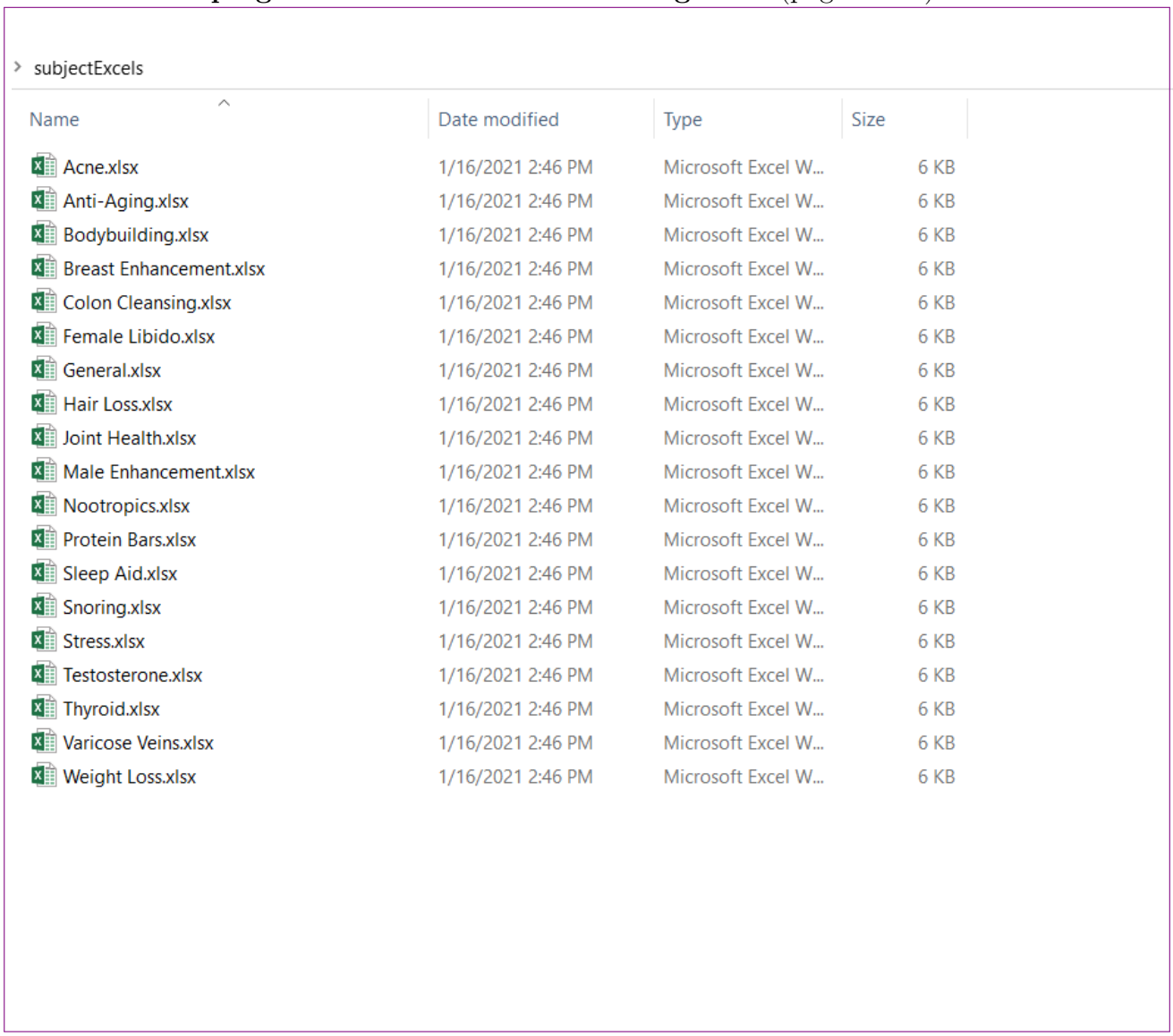
## BONUS - Scraping Bot - Logging In (page 5 of 8)



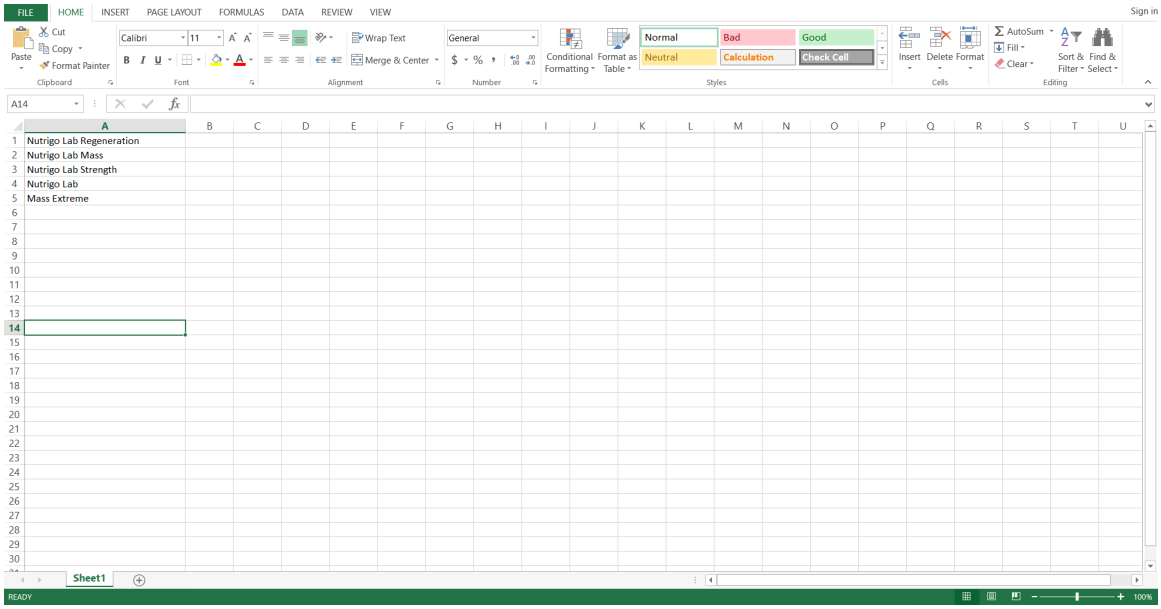
## BONUS - Scraping Bot - Viewing List of Products (page 6 of 8)



## BONUS - Scraping Bot - Folder After Collecting Data (page 7 of 8)



## **BONUS - Scraping Bot - Excel Example** (page 7 of 8)



## A.2 Săsăran Andrei Paul's code:

### A.2.1 Search

Code starts from the next page.

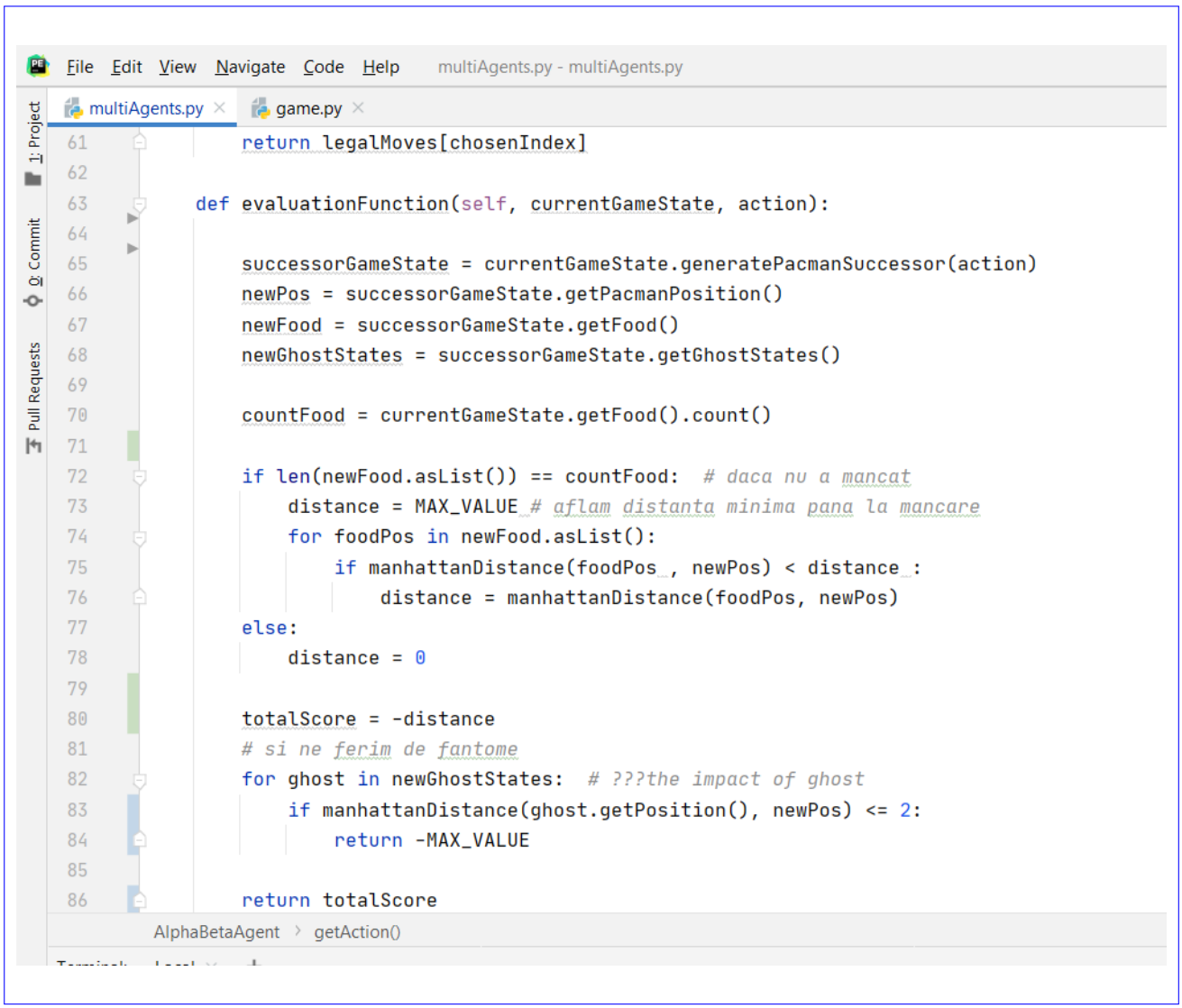
```
from Tkinter import *
import tkFont
import os
import sys
from model import *


from random import randint

def main():
    root = Tk()
    app = PuzzleWindow(root)


class PuzzleWindow:

    def __init__(self,master):
        self.master = master
        self.master.title("8 PUZZLE")
        self.master.geometry("220x400+900+200")
        self.master.resizable(0, 0)
        self.master.config(bg="#5DADE2")
        self.frame = Frame(self.master, background="#D6EAF8")
        self.frame.pack()

        self.goalWindow = Toplevel()
        self.goalWindow.geometry("250x300+1123+200")
        rootPath = os.path.dirname(os.path.abspath("goalState.gif"))
        rootPath = rootPath[:-4]
        rootPath = rootPath + "goalState.gif"
        goalImage = PhotoImage(file=rootPath)
        goalLabel = Label(self.goalWindow, image=goalImage)
        goalLabel.image = goalImage
        goalLabel.pack()
        self.goalWindow.overrideredirect(1)
        #.deiconify()



        shuffleFrame = LabelFrame(self.frame, width=300, height=50, background="#5DADE2")
        puzzleFrame = LabelFrame(self.frame, width=300, height=300, background="#2874A6")
        solutionFrame = LabelFrame(self.frame, width=300, height=120,
        background="#AED6F1")
        shuffleFrame.pack()
        puzzleFrame.pack()
        solutionFrame.pack()

        # SHUFFLE FRAME
        shuffleButton = Button(shuffleFrame,  text="SHUFFLE", command =
        self.shuffleFunction, bd=4, bg="#7FB3D5",width=10, height=1, font=tkFont.BOLD
        ).pack(side=LEFT)

        self.showGoal = IntVar(value=1)
        showGoalButton = Checkbutton(shuffleFrame, text="show GOAL",
        variable=self.showGoal, command = self.showGoalFunction, bd = 1, bg="#7FB3D5",
        width=10, height=1).pack(side=RIGHT)


        # PUZZLE FRAME
        self.puzzleCells = []
        #self.emptyCell = self.puzzleCells[0]
        noOfCells = 0

        # initialize puzzle
        for i in range (3):
            for j in range (3):
```

```
63                  cell = Label(puzzleFrame, text =
                    str(noOfCells),borderwidth=2,relief="groove",  width=5, height=3,
                    font=tkFont.BOLD, bg="#AED6F1")
64                  cell.grid(row = i, column = j)
65                  self.puzzleCells.append(cell)
66                  noOfCells = noOfCells + 1
67          self.puzzleCells[0].config(text = "")
68          self.emptyCellIndexGlobal = 0
69          self.inputPuzzle = []
70
71          # SOLUTION FRAME
72          solutionLabel = Label(solutionFrame, text = "SOLUTION:", font = tkFont.NORMAL,
            bg="#D4E6F1").pack(side=LEFT)
73          solveButton = Button(solutionFrame,  text="SOLVE", command =
            self.solveFunction, bd=4, bg="#7FB3D5",width=10, height=1, font=tkFont.BOLD
            ).pack(side=RIGHT)
74
75
76
77
78          self.master.mainloop()
79
80      # SHUFFLE FRAME
81      def showGoalFunction (self):
82          choose = self.showGoal.get()
83
84          if choose:
85              # publici fereastra
86              self.goalWindow.deiconify()
87
88          else:
89              # ascunzi fereastra
90              self.goalWindow.withdraw()
91
92      def shuffleFunction(self):
93          #reset the inputPuzzle
94          self.inputPuzzle = []
95
96          emptyCellIndex = self.emptyCellIndexGlobal
97          permutations = randint(10,30)
98          for i in range (0, permutations):
99              successors = self.getSuccessors(emptyCellIndex) # get successors
100             randomSuccessorIndex = successors[randint(0, len(successors)) - 1] # pick a
                random successor
101
102
103             currentLabel = self.puzzleCells[emptyCellIndex]
104             successorLabel = self.puzzleCells[randomSuccessorIndex]
105             successorValue = successorLabel['text']
106
107             currentLabel.configure(text=successorValue)
108             successorLabel.configure(text='')
109
110             emptyCellIndex = randomSuccessorIndex
111             self.emptyCellIndexGlobal = emptyCellIndex
112
113
114     # PUZZLE FRAME
115
116     def getSuccessors(self, cellIndex):
117         #cellIndex = self.puzzleCells.index(cell) # position of cell in puzzleCells list
118         row = cellIndex / 3 # position of cell in context
119         col = cellIndex % 3 # of matrix
120
121         successors = []
122
123         if (row > 0): # not on the first row
```

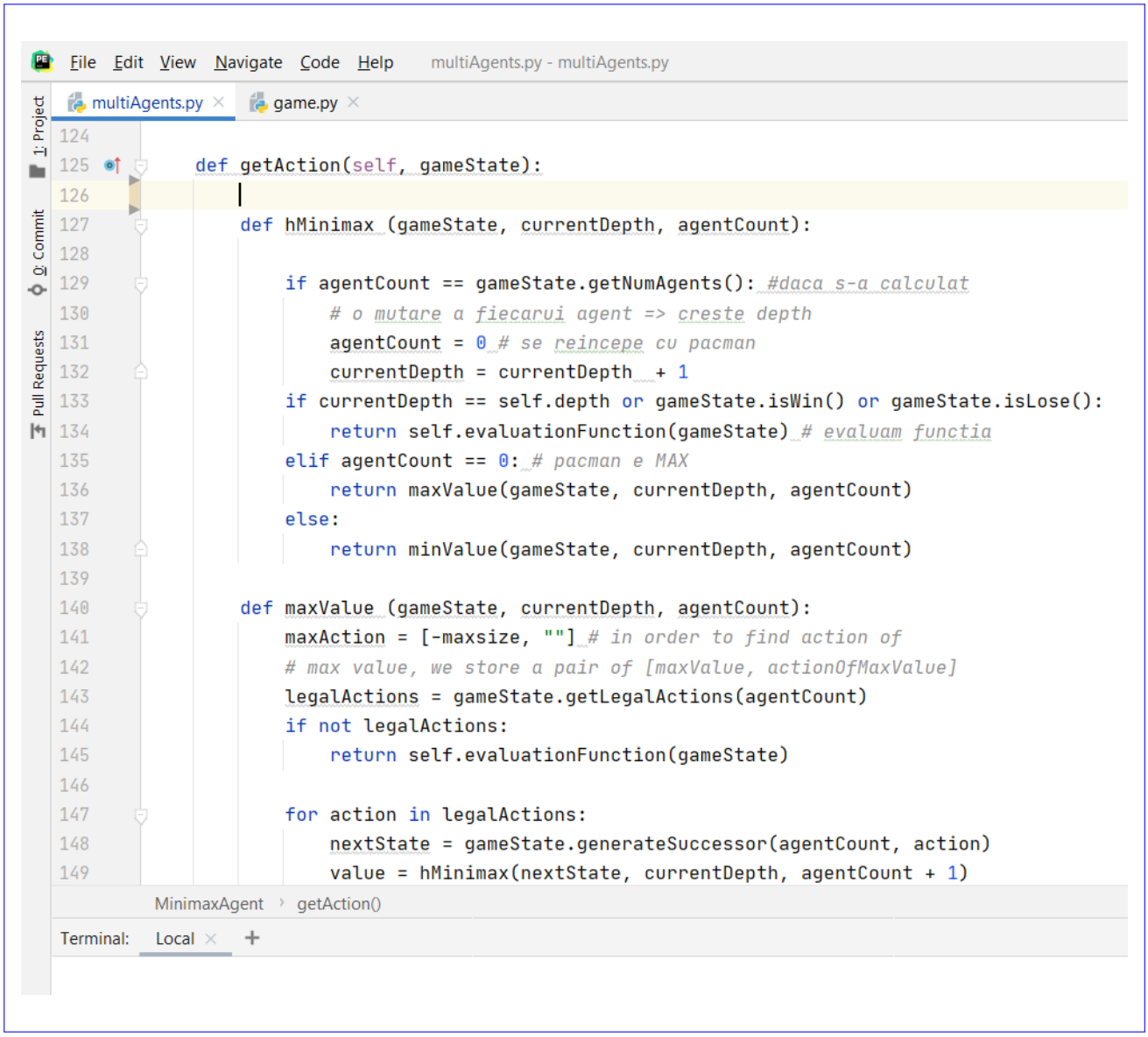
```
124                 successors.append(cellIndex - 3) # add the NORTH succesor
125             if (col < 2): # not on the right col
126                 successors.append(cellIndex + 1) # add the EAST succesor
127             if (row < 2): # not on the last row
128                 successors.append(cellIndex + 3) # add the SOUTH succesor
129             if (col > 0): # not on the left col
130                 successors.append(cellIndex - 1) # add the WEST succesor
131 
132             return successors
133 
134 
135 
136         def solveFunction(self):
137             self.transferPuzzle()
138             #print ("before")
139             #print self.inputPuzzle
140             self.model = PuzzleModel(self.inputPuzzle)
141             #print ("after")
142             #print self.inputPuzzle
143 
144             self.model.solveFunction()
145 
146 
147         def transferPuzzle (self):
148             count = 0
149             for i in range(0,3):
150                 for j in range(0,3):
151                     labelText = self.puzzleCells[count].cget("text")
152                     if (labelText  == ''):
153                         self.inputPuzzle.append(0)
154                     else:
155                         self.inputPuzzle.append(int(labelText))
156                     count = count + 1
157 
158 
159 
160 
161 if __name__ == '__main__':
162     main()
163 
164 
165 
```

# BONUS

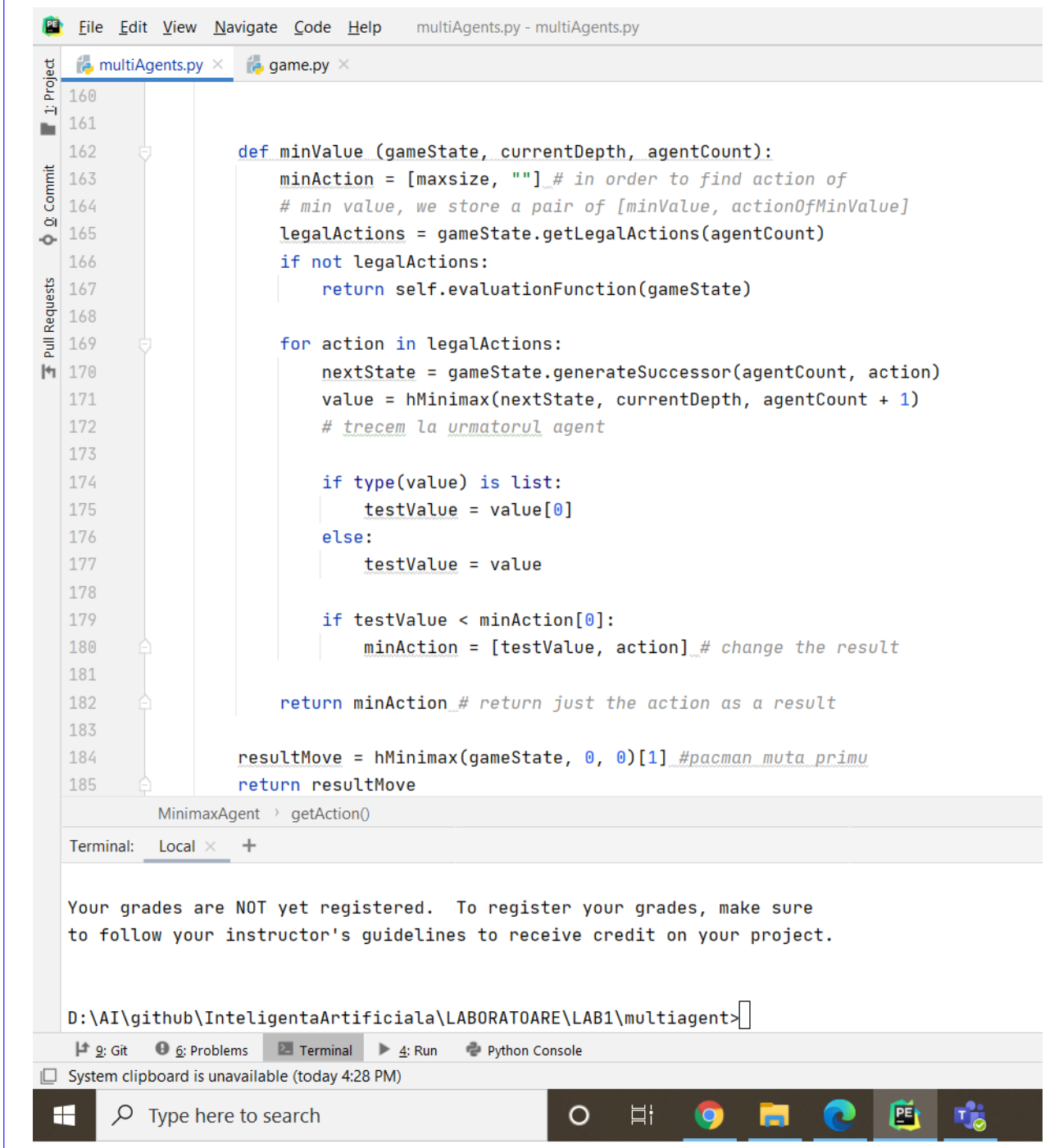
**Reflex Agent:** (page 1 of 1)



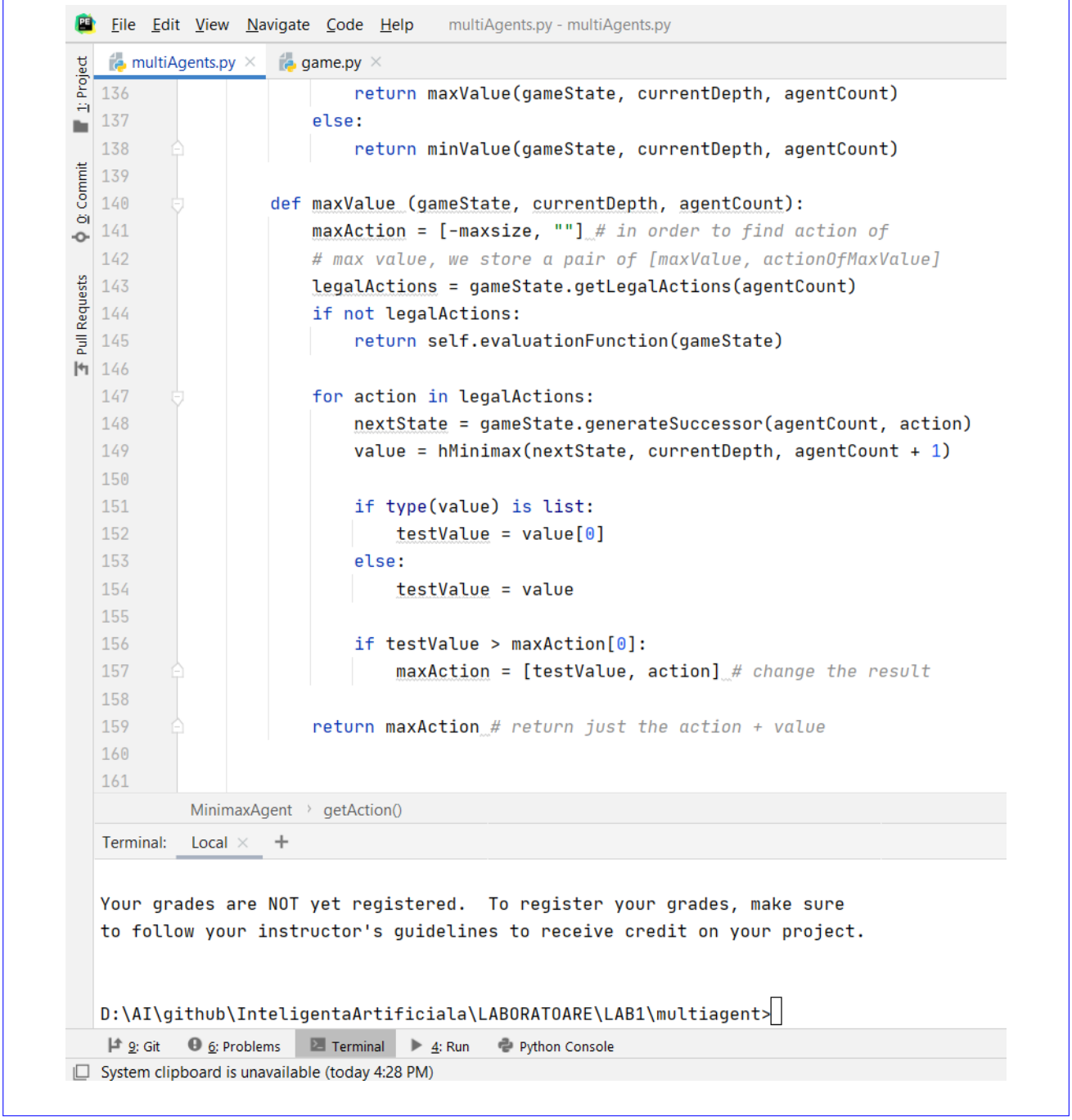
**Minimax Agent:** (page 1 of 3)



**Minimax Agent:** (page 2 of 3)



**Minimax Agent:** (page 3 of 3)



## A.2.2 Logics

**Who likes dolphins?** (page 1 of 4)

```
if(Mace4).
assign(domain_size, 4).
end_if.

if(Prover9).
formulas(goals).

HouseThree(h3) -> Canadian(h3).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% House number
    % HouseOne: you live in house one
    % HouseTwo: you live in house two
    % HouseThree: you live in house three
    % HouseFour: you live in house four

% Nationality
    % American: you are american
    % British: you are british
    % Canadian: you are canadian
    % Irish: you are irish

% Animal
    % Butterflies: you are the person with the butterflies
    % Dolphins: you are the person with dolphins
    % Horses: you are the person with the horses
    % Turtles: you are the person with the turtles

% Sport
    % Bowling: you play bowling
    % Handball: you play handball
    % Swimming: you play swimming
    % Tennis: you play tennis

% Colour
    % Black: the house is black
    % Blue: the house is blue
    % Red: the house is red
    % White: the house is white

formulas(assumptions).

% The three houses differ
    differentFrom(h1, h2).
    differentFrom(h1, h3).
    differentFrom(h1, h4).
    differentFrom(h2, h3).
    differentFrom(h2, h4).
    differentFrom(h3, h4).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    % Each house is assigned an order number, with reference to their position
    h1 = 0.
    h2 = 1.
    h3 = 2.
    h4 = 3.

    HouseOne(0).
    HouseTwo(1).
    HouseThree(2).
    HouseFour(3).

% Relation right(x, y) that affirms that "house y" is somewhere to the right of "house x"
```

```
    right(h1, h2).
    right(h1, h3).
    right(h1, h4).
    right(h2, h3).
    right(h2, h4).
    right(h3, h4).

    -right(h1, h1).
    -right(h2, h2).
    -right(h3, h3).
    -right(h4, h4).
    -right(h3, h1).
    -right(h4, h2).
    -right(h2, h1).
    -right(h3, h2).
    -right(h4, h3).
    -right(h4, h1).


% Relation rightneighbour(x, y) that affirms that "house y" is directly to the right of "house x"
    rightneighbour(h1, h2).
    rightneighbour(h2, h3).
    rightneighbour(h3, h4).

    -rightneighbour(h1, h1).
    -rightneighbour(h2, h2).
    -rightneighbour(h3, h3).
    -rightneighbour(h4, h4).
    -rightneighbour(h1, h3).
    -rightneighbour(h2, h4).
    -rightneighbour(h3, h1).
    -rightneighbour(h4, h2).
    -rightneighbour(h2, h1).
    -rightneighbour(h3, h2).
    -rightneighbour(h4, h3).
    -rightneighbour(h4, h1).
    -rightneighbour(h1, h4).

% Relation middleneighbour(x, y) that affirms that there is one house between "house x" and "house y"
    middleneighbour(h1, h3).
    middleneighbour(h3, h1).
    middleneighbour(h2, h4).
    middleneighbour(h4, h2).

    -middleneighbour(h1, h2).
    -middleneighbour(h2, h3).
    -middleneighbour(h3, h4).
    -middleneighbour(h1, h1).
    -middleneighbour(h2, h3).
    -middleneighbour(h3, h3).
    -middleneighbour(h4, h4).
    -middleneighbour(h2, h1).
    -middleneighbour(h3, h2).
    -middleneighbour(h4, h3).
    -middleneighbour(h4, h1).
    -middleneighbour(h1, h4).

% Relation inbetween(x, y) that affirms that there are two houses between "house x" and "house y"
    inbetween(h4, h1).
    inbetween(h1, h4).

    -inbetween(h1, h3).
    -inbetween(h3, h1).
    -inbetween(h2, h4).
    -inbetween(h4, h2).
    -inbetween(h1, h2).
```

**Who likes dolphins?** (page 3 of 4)

```
136       -inbetween(h2, h3).
137       -inbetween(h3, h4).
138       -inbetween(h1, h1).
139       -inbetween(h2, h3).
140       -inbetween(h3, h3).
141       -inbetween(h4, h4).
142       -inbetween(h2, h1).
143       -inbetween(h3, h2).
144       -inbetween(h4, h3).
145
146    % Each house is different and has a colour, nationality, animal and sport
147       HouseOne(x) | HouseTwo(x) | HouseThree(x) | HouseFour(x).
148       Black(x) | Blue(x) | Red(x) | White(x).
149       American(x) | British(x) | Canadian(x) | Irish(x).
150       Butterflies(x) | Dolphins(x) | Horses(x) | Turtles(x).
151       Bowling(x) | Handball(x) | Swimming(x) | Tennis(x).
152
153    % Each property applies at most to one house
154       Black(x) & Black(y) -> -differentFrom(x, y).
155       Blue(x) & Blue(y) -> -differentFrom(x, y).
156       Red(x) & Red(y) -> -differentFrom(x, y).
157       White(x) & White(y) -> -differentFrom(x, y).
158
159       American(x) & American(y) -> -differentFrom(x, y).
160       British(x) & British(y) -> -differentFrom(x, y).
161       Canadian(x) & Canadian(y) -> -differentFrom(x, y).
162       Irish(x) & Irish(y) -> -differentFrom(x, y).
163
164       Butterflies(x) & Butterflies(y) -> -differentFrom(x, y).
165       Dolphins(x) & Dolphins(y) -> -differentFrom(x, y).
166       Horses(x) & Horses(y) -> -differentFrom(x, y).
167       Turtles(x) & Turtles(y) -> -differentFrom(x, y).
168
169       Bowling(x) & Bowling(y) -> -differentFrom(x, y).
170       Handball(x) & Handball(y) -> -differentFrom(x, y).
171       Swimming(x) & Swimming(y) -> -differentFrom(x, y).
172       Tennis(x) & Tennis(y) -> -differentFrom(x, y).
173
174    %Clues
175
176       % 1. The second house is Black.
177       HouseTwo(x) -> Black(x).
178
179       % 2. The British lives in the first house.
180       HouseOne(x) -> British(x).
181
182       % 3. There are two houses between the person who likes Bowling and the person who
          likes Swimming.
183       Bowling(x) & Swimming(y) -> inbetween(x, y).
184
185       % 4. The American live directly to the left of the person who likes Turtles.
186       American(x) & Turtles(y) -> rightneighbour(x, y).
187
188       % 5. There are two houses between the person who likes Horses and the person who
          likes Butterflies on the right.
189       Horses(x) & Butterflies(y) -> inbetween(x, y).
190       HouseFour(x) -> Butterflies(x).
191
192       % 6. The person who likes Bowling lives somewhere to the right of the person who
          likes Tennis.
193       Bowling(x) & Tennis(y) -> right(y, x).
194
195       % 7. There is one house between the Irish and the person who likes Handball on the
          left.
196       Irish(x) & Handball(y) -> middleneighbour(x, y).
197       Irish(x) & Handball(y) -> right(y, x).
198
199       % 8. There is one house between the person who likes Horses and the Red house on
          the right.
```

```
200       Horses(x) & Red(y) -> middleneighbour(x, y).
201       Horses(x) & Red(y) -> right(x, y).
202
203       % 9. There is one house between the person who likes Handball and the White house
          on the right.
204       Handball(x) & White(y) -> middleneighbour(x, y).
205       Handball(x) & White(y) -> right(x, y).
206
207   end_of_list.
208
209
210
```

**Hardware Store** (page 1 of 5)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Man1(m1) -> Wrench(m1).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Men's order in row
    % Man1: first man
    % Man2: second man
    % Man3: third man
    % Man4: fourth man
    % Man5: fifth man

% Men's names
    % Billy, Dennis, George, Larry, Philip

% Shirt
    % Blue, Green, Red, White, Yellow

% Tool
    % Drill, Hammer, Saw, Screwdriver, Wrench

% Discount
    % FivePercent, TenPercent, FifteenPercent, TwentyPercent, TwentyFivePercent

% Profession
    % Electrician, Engineer, Mechanic, Plumber, Technician

% Age
    % ThirtyFive, Forty, FortyFive, Fifty, FiftyFive

formulas(assumptions).

% The 5 men differ
    differentFrom(m1, m2).
    differentFrom(m1, m3).
    differentFrom(m1, m4).
    differentFrom(m1, m5).
    differentFrom(m2, m3).
    differentFrom(m2, m4).
    differentFrom(m2, m5).
    differentFrom(m3, m4).
    differentFrom(m3, m5).
    differentFrom(m4, m5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    m1 = 0.
    m2 = 1.
    m3 = 2.
    m4 = 3.
    m5 = 4.

    Man1(0).
    Man2(1).
    Man3(2).
    Man4(3).
    Man5(4).

% Relation directlyright(x, y) that affirms that "man y" is directly to the right of
```

**Hardware Store** (page 2 of 5)

```
    "man x"
70      directlyright(m1, m2).
71      directlyright(m2, m3).
72      directlyright(m3, m4).
73      directlyright(m4, m5).
74
75      -directlyright(m1, m1).
76      -directlyright(m2, m2).
77      -directlyright(m3, m3).
78      -directlyright(m4, m4).
79      -directlyright(m5, m5).
80      -directlyright(m1, m3).
81      -directlyright(m2, m4).
82      -directlyright(m3, m5).
83      -directlyright(m3, m1).
84      -directlyright(m4, m2).
85      -directlyright(m5, m3).
86      -directlyright(m2, m1).
87      -directlyright(m3, m2).
88      -directlyright(m4, m3).
89      -directlyright(m5, m4).
90      -directlyright(m1, m4).
91      -directlyright(m2, m5).
92      -directlyright(m4, m1).
93      -directlyright(m5, m2).
94      -directlyright(m1, m5).
95      -directlyright(m5, m1).
96
97  % Relation nextto(x, y) that affirms that "man y" is next to "man x"
98      nextto(m1, m2).
99      nextto(m2, m1).
100     nextto(m2, m3).
101     nextto(m3, m2).
102     nextto(m3, m4).
103     nextto(m4, m3).
104     nextto(m4, m5).
105     nextto(m5, m4).
106
107     -nextto(m1, m1).
108     -nextto(m2, m2).
109     -nextto(m3, m3).
110     -nextto(m4, m4).
111     -nextto(m5, m5).
112     -nextto(m1, m3).
113     -nextto(m2, m4).
114     -nextto(m3, m5).
115     -nextto(m3, m1).
116     -nextto(m4, m2).
117     -nextto(m5, m3).
118     -nextto(m1, m4).
119     -nextto(m2, m5).
120     -nextto(m4, m1).
121     -nextto(m5, m2).
122     -nextto(m1, m5).
123     -nextto(m5, m1).
124
125 % Relation extremity(x) that affirms that "man x" is at one of the ends
126     extremity(m1).
127     extremity(m5).
128
129     -extremity(m2).
130     -extremity(m3).
131     -extremity(m4).
132
133 % Relation inbetween(x, y) that affirms that there is a man somewhere in between "man
    x" and "man y", exactly in that order
134     inbetween(m1, m3).
135     inbetween(m2, m4).
136     inbetween(m3, m5).
```

```
137       inbetween(m1, m4).
138       inbetween(m2, m5).
139       inbetween(m1, m5).
140
141       -inbetween(m1, m2).
142       -inbetween(m2, m1).
143       -inbetween(m2, m3).
144       -inbetween(m3, m2).
145       -inbetween(m3, m4).
146       -inbetween(m4, m3).
147       -inbetween(m4, m5).
148       -inbetween(m5, m4).
149       -inbetween(m1, m1).
150       -inbetween(m2, m2).
151       -inbetween(m3, m3).
152       -inbetween(m4, m4).
153       -inbetween(m5, m5).
154       -inbetween(m3, m1).
155       -inbetween(m4, m2).
156       -inbetween(m5, m3).
157       -inbetween(m4, m1).
158       -inbetween(m5, m2).
159       -inbetween(m5, m1).
160
161   % Relation right(x, y) that affirms that "man y" is somewhere to the right of "man x"
162       right(m1, m3).
163       right(m2, m4).
164       right(m3, m5).
165       right(m1, m4).
166       right(m2, m5).
167       right(m1, m5).
168       right(m1, m2).
169       right(m4, m5).
170       right(m2, m3).
171       right(m3, m4).
172
173       -right(m2, m1).
174       -right(m3, m2).
175       -right(m4, m3).
176       -right(m5, m4).
177       -right(m1, m1).
178       -right(m2, m2).
179       -right(m3, m3).
180       -right(m4, m4).
181       -right(m5, m5).
182       -right(m3, m1).
183       -right(m4, m2).
184       -right(m5, m3).
185       -right(m4, m1).
186       -right(m5, m2).
187       -right(m5, m1).
188
189   % Each man has different characteristics
190       Man1(x) | Man2(x) | Man3(x) | Man4(x) | Man5(x).
191       Billy(x) | Dennis(x) | George(x) | Larry(x) | Philip(x).
192       Blue(x) | Green(x) | Red(x) | White(x) | Yellow(x).
193       Drill(x) | Hammer(x) | Saw(x) | Screwdriver(x) | Wrench(x).
194       FivePercent(x) | TenPercent(x) | FifteenPercent(x) | TwentyPercent(x) |
          TwentyFivePercent(x).
195       Electrician(x) | Engineer(x) | Mechanic(x) | Plumber(x) | Technician(x).
196       ThirtyFive(x) | Forty(x) | FortyFive(x) | Fifty(x) | FiftyFive(x).
197
198   % Each property applies at most to one man
199       Billy(x) & Billy(y) -> -differentFrom(x, y).
200       Dennis(x) & Dennis(y) -> -differentFrom(x, y).
201       George(x) & George(y) -> -differentFrom(x, y).
202       Larry(x) & Larry(y) -> -differentFrom(x, y).
203       Philip(x) & Philip(y) -> -differentFrom(x, y).
204
```

```
205       Blue(x) & Blue(y) -> -differentFrom(x, y).
206       Green(x) & Green(y) -> -differentFrom(x, y).
207       Red(x) & Red(y) -> -differentFrom(x, y).
208       White(x) & White(y) -> -differentFrom(x, y).
209       Yellow(x) & Yellow(y) -> -differentFrom(x, y).
210
211       Drill(x) & Drill(y) -> -differentFrom(x, y).
212       Hammer(x) & Hammer(y) -> -differentFrom(x, y).
213       Saw(x) & Saw(y) -> -differentFrom(x, y).
214       Screwdriver(x) & Screwdriver(y) -> -differentFrom(x, y).
215       Wrench(x) & Wrench(y) -> -differentFrom(x, y).
216
217       FivePercent(x) & FivePercent(y) -> -differentFrom(x, y).
218       TenPercent(x) & TenPercent(y) -> -differentFrom(x, y).
219       FifteenPercent(x) & FifteenPercent(y) -> -differentFrom(x, y).
220       TwentyPercent(x) & TwentyPercent(y) -> -differentFrom(x, y).
221       TwentyFivePercent(x) & TwentyFivePercent(y) -> -differentFrom(x, y).
222
223       Electrician(x) & Electrician(y) -> -differentFrom(x, y).
224       Engineer(x) & Engineer(y) -> -differentFrom(x, y).
225       Mechanic(x) & Mechanic(y) -> -differentFrom(x, y).
226       Plumber(x) & Plumber(y) -> -differentFrom(x, y).
227       Technician(x) & Technician(y) -> -differentFrom(x, y).
228
229       ThirtyFive(x) & ThirtyFive(y) -> -differentFrom(x, y).
230       Forty(x) & Forty(y) -> -differentFrom(x, y).
231       FortyFive(x) & FortyFive(y) -> -differentFrom(x, y).
232       Fifty(x) & Fifty(y) -> -differentFrom(x, y).
233       FiftyFive(x) & FiftyFive(y) -> -differentFrom(x, y).
234
235   %Clues
236
237       % 1. The man wearing the Yellow shirt is somewhere to the left of the Mechanic.
238       Yellow(x) & Mechanic(y) -> right(x, y).
239
240       % 2. Billy is buying a tool with 20% off.
241       Billy(x) -> TwentyPercent(x).
242
243       % 3. The guy buying a tool with 15% discount is at one of the ends.
244       FifteenPercent(x) -> extremity(x).
245
246       % 4. The 45 years old man is somewhere between the Plumber and the 35 years old
          man, in that order.
247       Plumber(x) & ThirtyFive(y) -> inbetween(x, y).
248       FortyFive(x) & Plumber(y) -> right(y, x).
249       FortyFive(x) & ThirtyFive(y) -> right(x, y).
250
251       % 5. The one buying a tool with the biggest discount is next to the 50 years old man.
252       TwentyFivePercent(x) & Fifty(y) -> nextto(y, x).
253
254       % 6. Dennis is wearing the Green shirt.
255       Dennis(x) -> Green(x).
256
257       % 7. The 50 years old man is in the middle.
258       Man3(x) -> Fifty(x).
259
260       % 8. Larry is next to the man that is buying a tool with the smallest discount.
261       Larry(x) & TenPercent(y) -> nextto(y, x).
262
263       % 9. The one buying a Hammer is somewhere between the man buying the Wrench and the
          man buying the Screwdriver, in that order.
264       Wrench(x) & Screwdriver(y) -> inbetween(x, y).
265       Hammer(x) & Wrench(y) -> right(y, x).
266       Hammer(x) & Screwdriver(y) -> right(x, y).
267
268       % 10. The Technician is at the third position.
269       Man3(x) -> Technician(x).
270
271       % 11. The Plumber is next to the man wearing the Green shirt.
```

```
272       Plumber(x) & Green(y) -> nextto(y, x).
273
274       % 12. The oldest man is buying a Drill.
275       FiftyFive(x) -> Drill(x).
276
277       % 13. The guy wearing the White shirt is somewhere to the right of the man wearing
          the Blue shirt.
278       White(x) & Blue(y) -> right(y, x).
279
280       % 14. The man buying a Hammer is somewhere between the one who is getting 15%
          discount and the one buying a Saw, in that order.
281       FifteenPercent(x) & Saw(y) -> inbetween(x, y).
282       Hammer(x) & FifteenPercent(y) -> right(y, x).
283       Hammer(x) & Saw(y) -> right(x, y).
284
285       % 15. George is next to the Electrician.
286       George(x) & Electrician(y) -> nextto(y, x).
287
288       % 16. The man buying tools with 5% and 25% off are next to each other.
289       FivePercent(x) & TwentyFivePercent(y) -> nextto(x, y).
290       FivePercent(x) & TwentyFivePercent(y) -> nextto(y, x).
291
292       % 17. Philip is buying a Saw.
293       Philip(x) -> Saw(x).
294
295       % 18. At the second position is the man buying a Drill.
296       Man2(x) -> Drill(x).
297
298       % 19. The man wearing the Yellow shirt is next to the one buying a Saw.
299       Yellow(x) & Saw(y) -> nextto(y, x).
300
301       % 20. The guy getting the best discount is somewhere between the 55 years old man
          and the man buying a tool with 10% off, in that order.
302       FiftyFive(x) & TenPercent(y) -> inbetween(x, y).
303       TwentyFivePercent(x) & FiftyFive(y) -> right(y, x).
304       TwentyFivePercent(x) & TenPercent(y) -> right(x, y).
305
306       % 21. The man wearing the Red shirt is next to the man buying a tool with 15%
          discount.
307       Red(x) & FifteenPercent(y) -> nextto(y, x).
308
309   end_of_list.
310
```

**Black Friday** (page 1 of 5)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Man3(m3) -> Hank(m3).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z.

% Men's order in row
    % Man1: first man
    % Man2: second man
    % Man3: third man
    % Man4: fourth man
    % Man5: fifth man

% Men's names
    % Dustin, Eugene, Hank, Keith, Sean

% Shirt
    % Black, Blue, Green, Red, White

% Deal
    % BeardTrimmer, GameConsole, Laptop, Smartphone, TV

% Discount
    % FortyPercent, FiftyPercent, SixtyPercent, SeventyPercent, EightyPercent

% Age
    % TwentyFive, Thirty, ThirtyFive, Forty, FortyFive

% Juice
    % Apple, Cranberry, Grape, Lemon, Orange

formulas(assumptions).

% The 5 men differ
    differentFrom(m1, m2).
    differentFrom(m1, m3).
    differentFrom(m1, m4).
    differentFrom(m1, m5).
    differentFrom(m2, m3).
    differentFrom(m2, m4).
    differentFrom(m2, m5).
    differentFrom(m3, m4).
    differentFrom(m3, m5).
    differentFrom(m4, m5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    m1 = 0.
    m2 = 1.
    m3 = 2.
    m4 = 3.
    m5 = 4.

    Man1(0).
    Man2(1).
    Man3(2).
    Man4(3).
    Man5(4).

% Relation directlyright(x, y) that affirms that "man y" is directly to the right of
```

```
    "man x"
70      directlyright(m1, m2).
71      directlyright(m2, m3).
72      directlyright(m3, m4).
73      directlyright(m4, m5).
74
75      -directlyright(m1, m1).
76      -directlyright(m2, m2).
77      -directlyright(m3, m3).
78      -directlyright(m4, m4).
79      -directlyright(m5, m5).
80      -directlyright(m1, m3).
81      -directlyright(m2, m4).
82      -directlyright(m3, m5).
83      -directlyright(m3, m1).
84      -directlyright(m4, m2).
85      -directlyright(m5, m3).
86      -directlyright(m2, m1).
87      -directlyright(m3, m2).
88      -directlyright(m4, m3).
89      -directlyright(m5, m4).
90      -directlyright(m1, m4).
91      -directlyright(m2, m5).
92      -directlyright(m4, m1).
93      -directlyright(m5, m2).
94      -directlyright(m1, m5).
95      -directlyright(m5, m1).
96
97  % Relation nextto(x, y) that affirms that "man y" is next to "man x"
98      nextto(m1, m2).
99      nextto(m2, m1).
100     nextto(m2, m3).
101     nextto(m3, m2).
102     nextto(m3, m4).
103     nextto(m4, m3).
104     nextto(m4, m5).
105     nextto(m5, m4).
106
107     -nextto(m1, m1).
108     -nextto(m2, m2).
109     -nextto(m3, m3).
110     -nextto(m4, m4).
111     -nextto(m5, m5).
112     -nextto(m1, m3).
113     -nextto(m2, m4).
114     -nextto(m3, m5).
115     -nextto(m3, m1).
116     -nextto(m4, m2).
117     -nextto(m5, m3).
118     -nextto(m1, m4).
119     -nextto(m2, m5).
120     -nextto(m4, m1).
121     -nextto(m5, m2).
122     -nextto(m1, m5).
123     -nextto(m5, m1).
124
125 % Relation extremity(x) that affirms that "man x" is at one of the ends
126     extremity(m1).
127     extremity(m5).
128
129     -extremity(m2).
130     -extremity(m3).
131     -extremity(m4).
132
133 % Relation inbetween(x, y) that affirms that there is a man somewhere in between "man
    x" and "man y", exactly in that order
134     inbetween(m1, m3).
135     inbetween(m2, m4).
136     inbetween(m3, m5).
```

```
137       inbetween(m1, m4).
138       inbetween(m2, m5).
139       inbetween(m1, m5).
140
141       -inbetween(m1, m2).
142       -inbetween(m2, m1).
143       -inbetween(m2, m3).
144       -inbetween(m3, m2).
145       -inbetween(m3, m4).
146       -inbetween(m4, m3).
147       -inbetween(m4, m5).
148       -inbetween(m5, m4).
149       -inbetween(m1, m1).
150       -inbetween(m2, m2).
151       -inbetween(m3, m3).
152       -inbetween(m4, m4).
153       -inbetween(m5, m5).
154       -inbetween(m3, m1).
155       -inbetween(m4, m2).
156       -inbetween(m5, m3).
157       -inbetween(m4, m1).
158       -inbetween(m5, m2).
159       -inbetween(m5, m1).
160
161   % Relation right(x, y) that affirms that "man y" is somewhere to the right of "man x"
162       right(m1, m3).
163       right(m2, m4).
164       right(m3, m5).
165       right(m1, m4).
166       right(m2, m5).
167       right(m1, m5).
168       right(m1, m2).
169       right(m4, m5).
170       right(m2, m3).
171       right(m3, m4).
172
173       -right(m2, m1).
174       -right(m3, m2).
175       -right(m4, m3).
176       -right(m5, m4).
177       -right(m1, m1).
178       -right(m2, m2).
179       -right(m3, m3).
180       -right(m4, m4).
181       -right(m5, m5).
182       -right(m3, m1).
183       -right(m4, m2).
184       -right(m5, m3).
185       -right(m4, m1).
186       -right(m5, m2).
187       -right(m5, m1).
188
189   % Each man has different characteristics
190       Man1(x) | Man2(x) | Man3(x) | Man4(x) | Man5(x).
191       Dustin(x) | Eugene(x) | Hank(x) | Keith(x) | Sean(x).
192       Black(x) | Blue(x) | Green(x) | Red(x) | White(x).
193       BeardTrimmer(x) | GameConsole(x) | Laptop(x) | Smartphone(x) | TV(x).
194       FortyPercent(x) | FiftyPercent(x) | SixtyPercent(x) | SeventyPercent(x) |
          EightyPercent(x).
195       Apple(x) | Cranberry(x) | Grape(x) | Lemon(x) | Orange(x).
196       TwentyFive(x) | Thirty(x) | ThirtyFive(x) | Forty(x) | FortyFive(x).
197
198   % Each property applies at most to one man
199       Dustin(x) & Dustin(y) -> -differentFrom(x, y).
200       Eugene(x) & Eugene(y) -> -differentFrom(x, y).
201       Hank(x) & Hank(y) -> -differentFrom(x, y).
202       Keith(x) & Keith(y) -> -differentFrom(x, y).
203       Sean(x) & Sean(y) -> -differentFrom(x, y).
204
```

```
205       Black(x) & Black(y) -> -differentFrom(x, y).
206       Blue(x) & Blue(y) -> -differentFrom(x, y).
207       Green(x) & Green(y) -> -differentFrom(x, y).
208       Red(x) & Red(y) -> -differentFrom(x, y).
209       White(x) & White(y) -> -differentFrom(x, y).
210
211       BeardTrimmer(x) & BeardTrimmer(y) -> -differentFrom(x, y).
212       GameConsole(x) & GameConsole(y) -> -differentFrom(x, y).
213       Laptop(x) & Laptop(y) -> -differentFrom(x, y).
214       Smartphone(x) & Smartphone(y) -> -differentFrom(x, y).
215       TV(x) & TV(y) -> -differentFrom(x, y).
216
217       FortyPercent(x) & FortyPercent(y) -> -differentFrom(x, y).
218       FiftyPercent(x) & FiftyPercent(y) -> -differentFrom(x, y).
219       SixtyPercent(x) & SixtyPercent(y) -> -differentFrom(x, y).
220       SeventyPercent(x) & SeventyPercent(y) -> -differentFrom(x, y).
221       EightyPercent(x) & EightyPercent(y) -> -differentFrom(x, y).
222
223       Apple(x) & Apple(y) -> -differentFrom(x, y).
224       Cranberry(x) & Cranberry(y) -> -differentFrom(x, y).
225       Grape(x) & Grape(y) -> -differentFrom(x, y).
226       Lemon(x) & Lemon(y) -> -differentFrom(x, y).
227       Orange(x) & Orange(y) -> -differentFrom(x, y).
228
229       TwentyFive(x) & TwentyFive(y) -> -differentFrom(x, y).
230       Thirty(x) & Thirty(y) -> -differentFrom(x, y).
231       ThirtyFive(x) & ThirtyFive(y) -> -differentFrom(x, y).
232       Forty(x) & Forty(y) -> -differentFrom(x, y).
233       FortyFive(x) & FortyFive(y) -> -differentFrom(x, y).
234
235   %Clues
236
237       % 1. The man drinking the Orange juice is exactly to the right of the man who got
          the 70% discount.
238       Orange(x) & SeventyPercent(y) -> directlyright(y, x).
239
240       % 2. Keith is 45 years old.
241       Keith(x) -> FortyFive(x).
242
243       % 3. The man who bought the TV is exactly to the left of the man wearing the Red
          shirt.
244       TV(x) & Red(y) -> directlyright(x, y).
245
246       % 4. At the third position is the man who got the 50% discount.
247       Man3(x) -> FiftyPercent(x).
248
249       % 5. Keith is next to the man wearing the White shirt.
250       Keith(x) & White(y) -> nextto(y, x).
251
252       % 6. The 25-year-old man is somewhere between the 35-year-old man and the
          40-year-old man, in that order.
253       ThirtyFive(x) & Forty(y) -> inbetween(x, y).
254       TwentyFive(x) & ThirtyFive(y) -> right(y, x).
255       TwentyFive(x) & Forty(y) -> right(x, y).
256
257       % 7. The man drinking Apple juice bought the Smartphone.
258       Apple(x) -> Smartphone(x).
259
260       % 8. The 30-year-old man is exactly to the left of the man that bought the Beard
          trimer.
261       Thirty(x) & BeardTrimmer(y) -> directlyright(x, y).
262
263       % 9. Sean is the youngest.
264       Sean(x) -> TwentyFive(x).
265
266       % 10. The man that got the 40% discount is exactly to the right of the man who
          bought the Beard trimer.
267       FortyPercent(x) & BeardTrimmer(y) -> directlyright(y, x).
268
```

```
269       % 11. Keith is next to the 35-year-old man.
270       Keith(x) & ThirtyFive(y) -> nextto(y, x).
271
272       % 12. Eugene is 40 years old.
273       Eugene(x) -> Forty(x).
274
275       % 13. Sean is wearing the Black shirt.
276       Sean(x) -> Black(x).
277
278       % 14. At the fourth position is the man who got the biggest discount.
279       Man4(x) -> EightyPercent(x).
280
281       % 15. Dustin got 60% off.
282       Dustin(x) -> SixtyPercent(x).
283
284       % 16. The man drinking the Lemon juice is exactly to the right of the man drinking
          the Grape juice.
285       Lemon(x) & Grape(y) -> directlyright(y, x).
286
287       % 17. Keith bought a Game console.
288       Keith(x) -> GameConsole(x).
289
290       % 18. The man who got the 80% discount is exactly to the left of the man who is
          wearing the Blue shirt.
291       EightyPercent(x) & Blue(y) -> directlyright(x, y).
292
293       % 19. The man drinking Grape juice bought the Beard trimer.
294       Grape(x) -> BeardTrimmer(x).
295
296       % 20. The man wearing the Black shirt is somewhere to the right of Keith.
297       Black(x) & Keith(y) -> right(y, x).
298
299       % 21. The man that bought the Smartphone is next to the man wearing the Black shirt.
300       Smartphone(x) & Black(y) -> nextto(y, x).
301
302   end_of_list.
303
```

**Home Remodeling** (page 1 of 4)

```
if(Mace4).
assign(domain_size, 5).
end_if.

if(Prover9).
formulas(goals).

Couple1(d1) -> Stephen(d1).

end_of_list.
end_if.

% Avoid using sentence names starting with u, v, w, x, y, z

% Couple's order in row
    % Couple1: first man
    % Couple2: second man
    % Couple3: third man
    % Couple4: fourth man
    % Couple5: fifth man

% Carts' colors
    % Black, Blue, Green, Red, White

% Husbands
    % Benny, Darrel, Myles, Stephen, Vince

% Wifes
    % Amelia, Jeanne, Kassie, Sylvia, Toni

% Room
    % Bathroom, Bedroom, DiningRoom, Kitchen, LivingRoom

% Budget
    % ThreeThousand, SixThousand, NineThousand, TwelveThousand, FifteenThousand

% Inspiration
    % Exhibition, FriendsHouse, Magazine, TVshow, Website

formulas(assumptions).

% The 5 couples differ
    differentFrom(d1, d2).
    differentFrom(d1, d3).
    differentFrom(d1, d4).
    differentFrom(d1, d5).
    differentFrom(d2, d3).
    differentFrom(d2, d4).
    differentFrom(d2, d5).
    differentFrom(d3, d4).
    differentFrom(d3, d5).
    differentFrom(d4, d5).

% The "differentFrom" relation is symmetrical
    differentFrom(x, y) -> differentFrom(y, x).

    d1 = 0.
    d2 = 1.
    d3 = 2.
    d4 = 3.
    d5 = 4.

    Couple1(0).
    Couple2(1).
    Couple3(2).
    Couple4(3).
    Couple5(4).

% Relation directlyright(x, y) that affirms that "couple y" is directly to the right of
```

```
    "couple x"
70      directlyright(d1, d2).
71      directlyright(d2, d3).
72      directlyright(d3, d4).
73      directlyright(d4, d5).
74
75      -directlyright(d1, d1).
76      -directlyright(d2, d2).
77      -directlyright(d3, d3).
78      -directlyright(d4, d4).
79      -directlyright(d5, d5).
80      -directlyright(d1, d3).
81      -directlyright(d2, d4).
82      -directlyright(d3, d5).
83      -directlyright(d3, d1).
84      -directlyright(d4, d2).
85      -directlyright(d5, d3).
86      -directlyright(d2, d1).
87      -directlyright(d3, d2).
88      -directlyright(d4, d3).
89      -directlyright(d5, d4).
90      -directlyright(d1, d4).
91      -directlyright(d2, d5).
92      -directlyright(d4, d1).
93      -directlyright(d5, d2).
94      -directlyright(d1, d5).
95      -directlyright(d5, d1).
96
97  % Relation nextto(x, y) that affirms that "couple y" is next to "couple x"
98      directlyright(x, y) | directlyright(y, x) <-> nextto(x, y).
99
100 % Relation extremity(x) that affirms that "couple x" is at One of the ends.
101     extremity(d1).
102     extremity(d5).
103
104     -extremity(d2).
105     -extremity(d3).
106     -extremity(d4).
107
108 % Relation right(x, y) that affirms that "couple y" is somewhere to the right of
    "couple x"
109     right(d1, d3).
110     right(d2, d4).
111     right(d3, d5).
112     right(d1, d4).
113     right(d2, d5).
114     right(d1, d5).
115     right(d1, d2).
116     right(d4, d5).
117     right(d2, d3).
118     right(d3, d4).
119
120     -right(d2, d1).
121     -right(d3, d2).
122     -right(d4, d3).
123     -right(d5, d4).
124     -right(d1, d1).
125     -right(d2, d2).
126     -right(d3, d3).
127     -right(d4, d4).
128     -right(d5, d5).
129     -right(d3, d1).
130     -right(d4, d2).
131     -right(d5, d3).
132     -right(d4, d1).
133     -right(d5, d2).
134     -right(d5, d1).
135
136 % % Each couple has different characteristics
```

```
137       Couple1(x) | Couple2(x) | Couple3(x) | Couple4(x) | Couple5(x).
138       Benny(x) | Darrel(x) | Myles(x) | Stephen(x) | Vince(x).
139       Black(x) | Blue(x) | Green(x) | Red(x) | White(x).
140       Amelia(x) | Jeanne(x) | Kassie(x) | Sylvia(x) | Toni(x).
141       ThreeThousand(x) | SixThousand(x) | NineThousand(x) | TwelveThousand(x) |
          FifteenThousand(x).
142       Exhibition(x) | FriendsHouse(x) | Magazine(x) | TVshow(x) | Website(x).
143       Bathroom(x) | Bedroom(x) | DiningRoom(x) | Kitchen(x) | LivingRoom(x).
144
145   % Each property applies at most to one couple
146       Benny(x) & Benny(y) -> -differentFrom(x, y).
147       Darrel(x) & Darrel(y) -> -differentFrom(x, y).
148       Myles(x) & Myles(y) -> -differentFrom(x, y).
149       Stephen(x) & Stephen(y) -> -differentFrom(x, y).
150       Vince(x) & Vince(y) -> -differentFrom(x, y).
151
152       Black(x) & Black(y) -> -differentFrom(x, y).
153       Blue(x) & Blue(y) -> -differentFrom(x, y).
154       Green(x) & Green(y) -> -differentFrom(x, y).
155       Red(x) & Red(y) -> -differentFrom(x, y).
156       White(x) & White(y) -> -differentFrom(x, y).
157
158       Amelia(x) & Amelia(y) -> -differentFrom(x, y).
159       Jeanne(x) & Jeanne(y) -> -differentFrom(x, y).
160       Kassie(x) & Kassie(y) -> -differentFrom(x, y).
161       Sylvia(x) & Sylvia(y) -> -differentFrom(x, y).
162       Toni(x) & Toni(y) -> -differentFrom(x, y).
163
164       ThreeThousand(x) & ThreeThousand(y) -> -differentFrom(x, y).
165       SixThousand(x) & SixThousand(y) -> -differentFrom(x, y).
166       NineThousand(x) & NineThousand(y) -> -differentFrom(x, y).
167       TwelveThousand(x) & TwelveThousand(y) -> -differentFrom(x, y).
168       FifteenThousand(x) & FifteenThousand(y) -> -differentFrom(x, y).
169
170       Exhibition(x) & Exhibition(y) -> -differentFrom(x, y).
171       FriendsHouse(x) & FriendsHouse(y) -> -differentFrom(x, y).
172       Magazine(x) & Magazine(y) -> -differentFrom(x, y).
173       TVshow(x) & TVshow(y) -> -differentFrom(x, y).
174       Website(x) & Website(y) -> -differentFrom(x, y).
175
176       Bathroom(x) & Bathroom(y) -> -differentFrom(x, y).
177       Bedroom(x) & Bedroom(y) -> -differentFrom(x, y).
178       DiningRoom(x) & DiningRoom(y) -> -differentFrom(x, y).
179       Kitchen(x) & Kitchen(y) -> -differentFrom(x, y).
180       LivingRoom(x) & LivingRoom(y) -> -differentFrom(x, y).
181
182   %Clues
183
184       % 1. The couple guiding the Black cart is somewhere between the couple inspired by
          a Website and the couple guiding the Red cart, in that order.
185       Website(x) & Black(y) & Red(z) -> right(x, y) & right(y, z).
186
187       % 2. Amelia has a $ 15,000 budget.
188       Amelia(x) -> FifteenThousand(x).
189
190       % 3. The partners who will remodel their Bedroom is next to the couple who will
          remodel their Living room.
191       Bedroom(x) & LivingRoom(y) -> nextto(y, x).
192
193       % 4. Kassie is at one of the ends.
194       Kassie(x) -> extremity(x).
195
196       % 5. Benny is next to the partners steering the Black shopping cart.
197       Benny(x) & Black(y) -> nextto(y, x).
198
199       % 6. At the first position is the couple who has the biggest budget.
200       Couple1(x) -> FifteenThousand(x).
201
202       % 7. The couple inspired by a Magazine is exactly to the right of the couple that
```

```
        will remodel their Living room.
203     Magazine(x) & LivingRoom(y) -> directlyright(y, x).
204
205     % 8. Jeanne is immediately before the partners that are going to remodel their
        Bathroom.
206     Jeanne(x) & Bathroom(y) -> directlyright(x, y).
207
208     % 9. Toni was inspired by a interior design Exhibition.
209     Toni(x) -> Exhibition(x).
210
211     % 10. The couple guiding the Green cart is going to remodel their Bathroom.
212     Green(x) -> Bathroom(x).
213
214     % 11. At the fifth position is the couple who got an inspiration from a Friend's
        house.
215     Couple5(x) -> FriendsHouse(x).
216
217     % 12. Benny is next to Sylvia and her husband.
218     Benny(x) & Sylvia(y) -> nextto(y, x).
219
220     % 13. Vince is exactly to the right of the couple that will remodel their Kitchen.
221     Vince(x) & Kitchen(y) -> directlyright(y, x).
222
223     % 14. Myles and his wife will remodel their Bathroom.
224     Myles(x) -> Bathroom(x).
225
226     % 15. The couple inspired by a TV show is somewhere between the couple that has the
        smallest budget and the couple that has a $ 9,000 budget, in that order.
227     ThreeThousand(x) & TVshow(y) & NineThousand(z) -> right(x, y) & right(y, z).
228
229     % 16. The partners steering the the Red cart is somewhere to the left of the couple
        steering the White cart.
230     Red(x) & White(y) -> right(x, y).
231
232     % 17. The couple that will remodel their Kitchen is immediately before the couple
        that will remodel their Bedroom.
233     Kitchen(x) & Bedroom(y) -> directlyright(x, y).
234
235     % 18. The partners that have a $ 12,000 budget is somewhere to the right of the
        partners guiding the Green cart.
236     TwelveThousand(x) & Green(y) -> right(y, x).
237
238     % 19. Darrel is exactly to the right of the couple who has a $ 9,000 budget.
239     Darrel(x) & NineThousand(y) -> directlyright(y, x).
240
241
242 end_of_list.
243
```

### A.2.3 Planning

**Clinical Trial - Domain** (page 1 of 2)

```
(define (domain clinical-trials)
(:predicates (is-candidate ?p)
             (healthy ?p)
             (blood-tested ?p)
             (is-selected ?p)
             (additional-tested ?p))

   (:action senseADDITIONALTESTS
   :parameters (?p)
   :observe (additional-tested ?p))

   (:action make-blood-tests
   :parameters (?p)
   :precondition (and (is-candidate ?p) (additional-tested ?p))
   :effect (and (nondet (healthy ?p)) (blood-tested ?p)))

   (:action make-more-tests
   :parameters (?p)
   :precondition (and (is-candidate ?p) (not (additional-tested ?p)))
   :effect (and (additional-tested ?p)))

   (:action select-candidate
   :parameters (?p)
   :precondition (and (healthy ?p) (is-candidate ?p) (blood-tested ?p))
   :effect (and (is-selected ?p)))

   (:action give-medication
   :parameters (?p)
   :precondition (and (blood-tested ?p))
   :effect (and (when (not (healthy ?p)) (healthy ?p))))
)

```

**Clinical Trial - Problem** (page 2 of 2)

```
(define (problem clinical-trial-selecting-phase)
    (:domain clinical-trial)
    (:objects candidate1 candidate2 candidate3 candidate4)
    (:init
        (is-candidate candidate1)
        (is-candidate candidate2)
        (is-candidate candidate3)
        (is-candidate candidate4)

        (additional-tested candidate3)
        (additional-tested candidate4)

        (unknown (additional-tested candidate1))
        (unknown (additional-tested candidate2))

        (oneof (additional-tested candidate1)
               (additional-tested candidate2)))

    (:goal (and (is-selected candidate1) (is-selected candidate3) (is-selected
    candidate4) (is-selected candidate2))))
```

**Sliding Tiles - Problem** (page 1 of 1)

```
(define (problem eight-puzzle)
    (:domain sliding-tile)
    (:objects
        tile1 tile2 tile3 tile4 tile5 tile6 tile7 tile8
        row1 row2 row3
        col1 col2 col3)
    (:init
        (next-row row1 row2)       (next-col col1 col2)
        (next-row row2 row3)       (next-col col2 col3)
        (tile-at tile8 row2 col2)  (tile-at tile4 row2 col1)
        (tile-at tile7 row3 col1)  (tile-at tile3 row1 col3)
        (tile-at tile6 row3 col2)  (tile-at tile2 row1 col2)
        (tile-at tile5 row2 col3)  (tile-at tile1 row1 col1)
        (is-blank row3 col3))
    (:goal
       (and
        (tile-at tile1 row1 col1)  (tile-at tile2 row1 col2)
        (tile-at tile3 row1 col3)  (tile-at tile4 row2 col1)
        (tile-at tile5 row2 col2)  (tile-at tile6 row2 col3)
        (tile-at tile7 row3 col1)  (tile-at tile8 row3 col2)))
        )


```

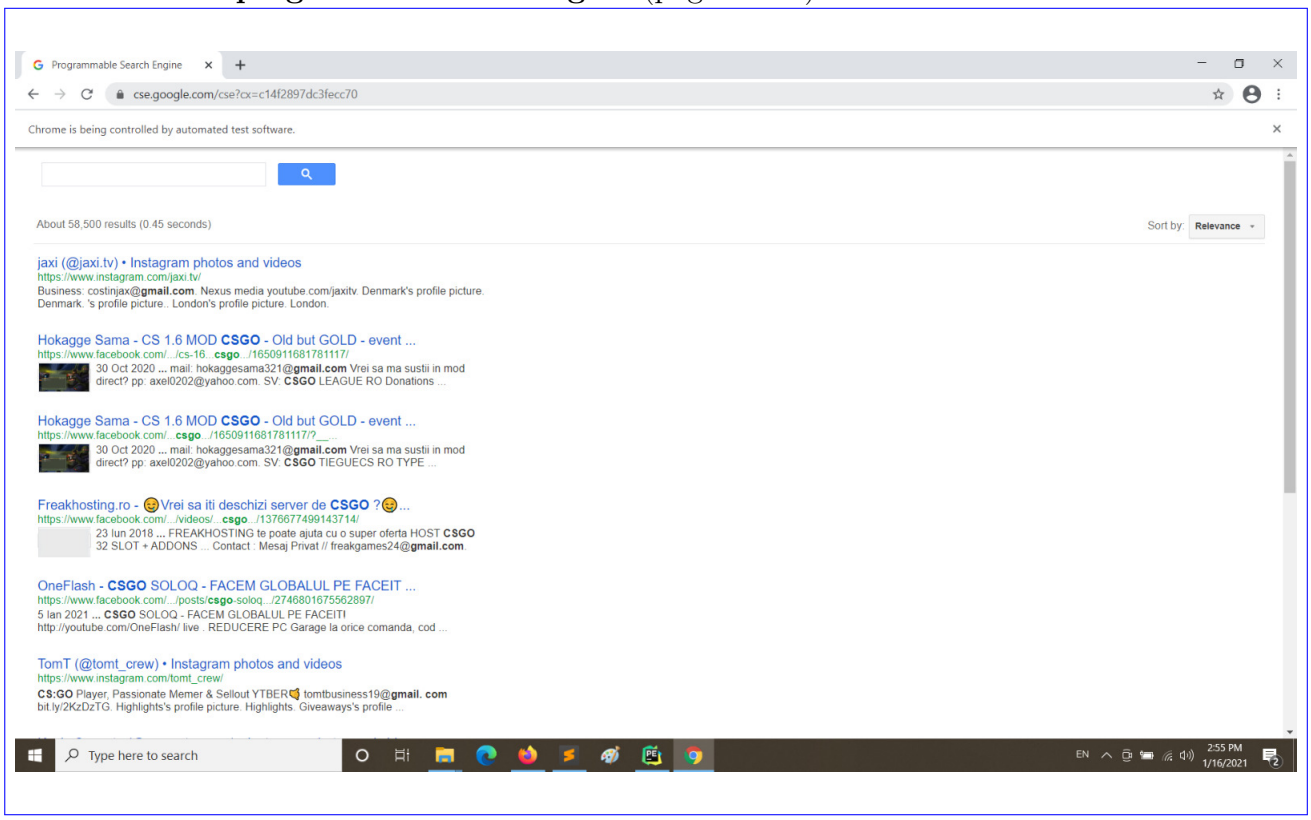
## BONUS - Scraping Bot - Code (page 1 of 7)

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.select import Select

import unicodedata

from selenium.webdriver.common.action_chains import ActionChains
import time

PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)

GOOGLE_ENGINE_LINK = "https://cse.google.com/cse?cx=c14f2897dc3fecc70"
LITE_LINK = "https://www.lite14.us/"
SEARCH_BAR_MESSAGE = "csgo @gmail.com"
SEARCH_BAR_ID = "gsc-i-id1"
CSE_HEADER_ID = "cse-header"
LITE_COOKIES_XPATH = "/html/body/div/div/a"
LITE_INPUT_TEXT_ID = "myInput"
LITE_COMBOBOX_XPATH =
"/html/body/table/tbody/tr/td[2]/div/div/table/tbody/tr/td[1]/form/table/tbody/tr/td/tabl
e/tbody/tr[3]/td/select"


ALL_MAIL_DATA = ""
seenCookies = False

def openNewTabWithPage (resultPage, driver):
    driver.execute_script("window.open('');")
    driver.switch_to.window(driver.window_handles[1])
    driver.get()
    pageNumber = str(resultPage.text)
    pageLink = driver.find_element_by_link_text(pageNumber)
    pageLink.click()

def copyPasteData (driver, searchBar): # copy the data and put it in search bar, then
return as a beauty string (normalized)
    copyPasteAll = ActionChains(driver)
    copyPasteAll.key_down(Keys.CONTROL).send_keys("a").perform() # perform a control + A
    copyPasteAll.key_down(Keys.CONTROL).send_keys("c").perform() # perform a control + C
    copyPasteAll.click(searchBar).perform() # dam click pe searchBar
    copyPasteAll.key_down(Keys.CONTROL).send_keys("v").perform() # perform a control + V

    dataText = searchBar.get_attribute('value') # store data in dataText
    inputTextNormalized = unicodedata.normalize('NFKD',
    dataText).encode('ascii','ignore') # il normalizam pentru a l pune in textarea
    return inputTextNormalized

def convertDataToMails (dataToBeConverted, seenCookies):
    # ACUM DESCHIDEM UN TAB NOU CU LITE CA SA EXTRAGEM MAILURILE
    driver.execute_script("window.open('');")
    driver.switch_to.window(driver.window_handles[1])
    driver.get(LITE_LINK)

    if (seenCookies == False): # daca suntem prima oara cand accesam siteu
        # tre sa apasam pe cookies altfel nu
        time.sleep(7) # wait pana apare ala de cookies
        cookies = driver.find_element_by_xpath(LITE_COOKIES_XPATH)
        cookies.click()
    else:
        time.sleep(7)

    liteInput = driver.find_element_by_id(LITE_INPUT_TEXT_ID)
    liteInput.click()
    liteInput.send_keys(Keys.TAB) # punem un tab si il stergem
    liteInput.clear() # doar asa mere (asa am gasit pe net)
    liteInput.send_keys(dataToBeConverted) # si trimitem fisierul brut pentru a scoate
```

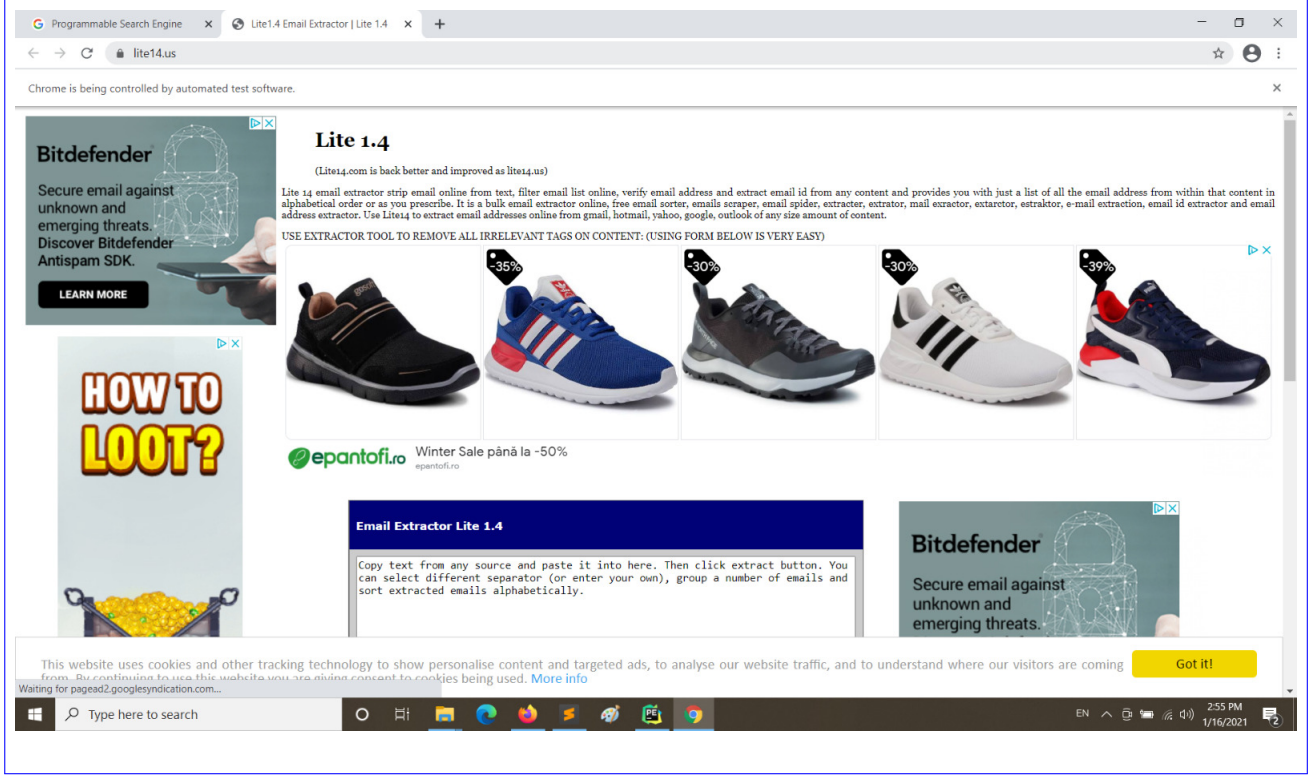
```
        email-urile
64
65      select = Select(driver.find_element_by_xpath(LITE_COMBOBOX_XPATH)) # selectam
        separatorul
66      select.select_by_value("new") # la newLine
67
68      extractButton =
        driver.find_element_by_xpath("/html/body/table/tbody/tr/td[2]/div/div/table/tbody/tr/
        td[1]/form/table/tbody/tr/td/table/tbody/tr[4]/td[1]/input[1]")
69      extractButton.click() # BOOM ACUMA AVEM MAILURILE. tre sa le stocam undeva
70
71      time.sleep(3)
72      mails = liteInput.get_attribute('value')
73
74      mailsList = mails.split("\n")
75      #for mail in mailsList:
76          #print str(mail)
77      return mailsList
78
79
80  driver.get(GOOGLE_ENGINE_LINK)
81
82  searchBar = driver.find_element_by_id(SEARCH_BAR_ID) # cautam searchBarul
83  searchBar.send_keys(SEARCH_BAR_MESSAGE) # ii trimitem mesajul
84  searchBar.send_keys(Keys.RETURN) # RETURN = ENTER
85  searchBar.clear() # curatam searchBarul (pt PASTE)
86
87  cseHeader = driver.find_element_by_id(CSE_HEADER_ID) # dam click undeva ca
88  cseHeader.click() # sa putem da Ctrl+A
89  ## incerc sa trec prin toate paginile
90  time.sleep(2) # asteptam sa se incarce pagina ca sa nu dam Ctrl+A la nimic
91
92
93  resultPages = driver.find_elements_by_class_name("gsc-cursor-page") # get all the
    searches pages
94  #print len(resultPages)
95
96  print "before:"
97  print ALL_MAIL_DATA
98
99  ALL_MAIL = []
100
101 #for i in range (1,len(resultPages)): # mergem pe fiecare pagina
102 for i in range (1,4): # mergem pe fiecare pagina
103     resultPages = driver.find_elements_by_class_name("gsc-cursor-page") # recalculam
        pentru ca is sanse sa dispara
104     time.sleep(3)
105     currentPage = resultPages[i] # selectam pagina curenta
106     currentData = copyPasteData(driver, searchBar) # nu putem salva toata RAW_DATA
107     # de asta facem conversia si salvam numa ce am convertit
108     ALL_MAIL.append(convertDataToMails(currentData, seenCookies)) # asta schimba tabu
109     seenCookies = True
110
111     # TO DO CHANGE THE TAB
112     driver.switch_to.window(driver.current_window_handle)
113     driver.close()
114     driver.switch_to.window(driver.window_handles[0])
115     time.sleep(5) # wait to close current tab
116     currentPage.click() # trecem la urmatoarea pagina de Search neexplorata
117     time.sleep(3)
118
119 print "after:"
120 setMail = set()
121 for mail in ALL_MAIL:
122     setMail.add(str(mail))
123
124 for mail in setMail:
```

```
125         print mail
126 
127     time.sleep(60) # delay ca sa putemm vedea ce face asta
128     driver.quit()
```

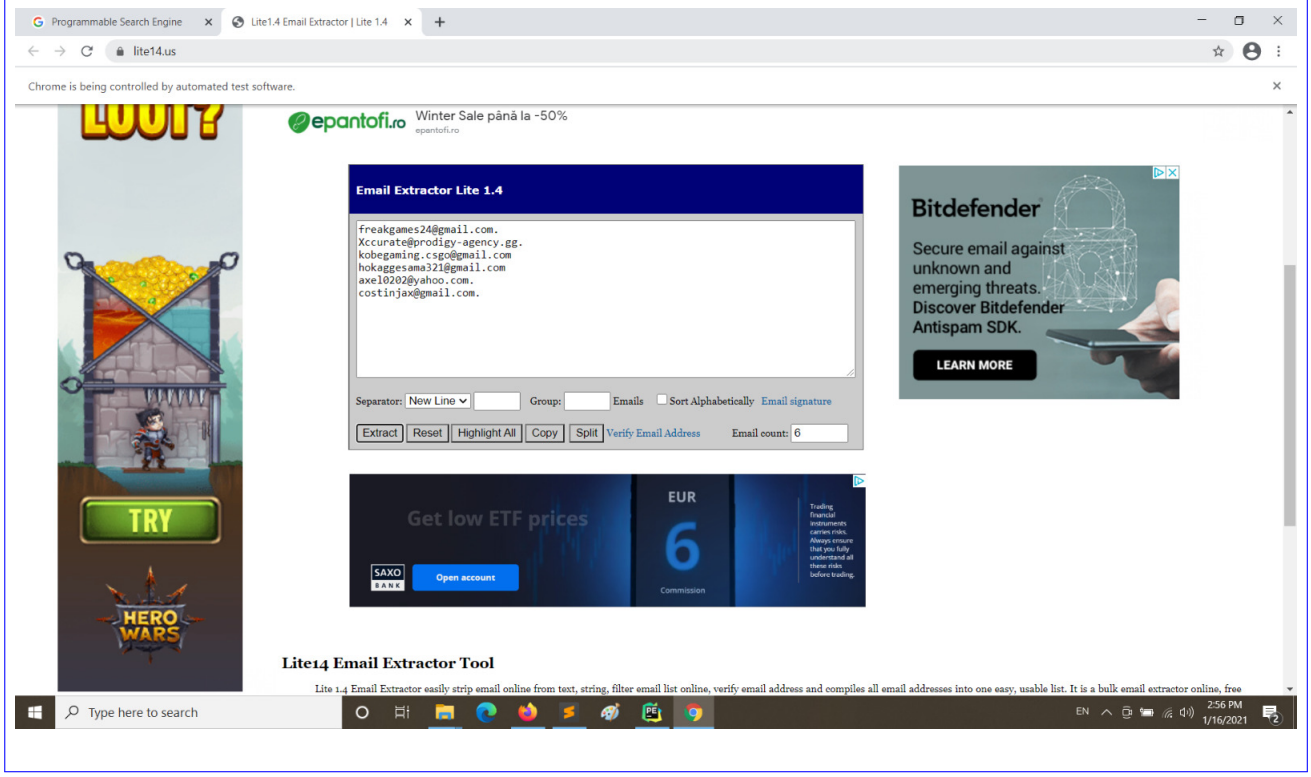
## **BONUS - Scraping Bot - Search Engine** (page 4 of 7)



## BONUS - Scraping Bot - Before Extracting Mails (page 5 of 7)



## BONUS - Scraping Bot - After Extracting Mails (page 6 of 7)



## BONUS - Scraping Bot - Console Output (page 7 of 7)

```
before:

after:
[u'freakgames24@gmail.com.', u'Xccurate@prodigy-agency.gg.', u'kobegaming.csgo@gmail.com', u'hokaggesama321@gmail.com', u'axel0202@yahoo.com.', u'costinjax@gmail.com.']
```