Middle East Technical University, NCC

CNG111 - Introduction to Computer Engineering Concepts

Fall 2014 - 2015

Assignment II - Secret Code Comprehension - 100 pts

Due Date: 28.12.14, midnight

For this assignment, you will work on a code breaking game between an **Agent** and a **CodeBreaker**. The game proceeds as follows:

1. The **Agent** picks a random password.
   - The password must contain 4 unique digits and it should not start with the digit `0`.
   - **Example:** `0123` is invalid password since it starts with a `0`
   - **Example:** `1231` is invalid because it contains multiple `1`s.
   - **Example:** `980` is invalid because it does not contain 4 digits.
   - **Example:** `1234, 3154, 9032` are all valid passwords.
2. In an attempt to find the password, **CodeBreaker** picks a **Guess**, which is also a 4 digit number with all unique digits and a non-zero first digit. The examples for valid/invalid passwords above also directly apply to the **Guess**.
3. The **Agent** looks at the **Guess** and reports back two numbers:
   (a) **Matches:** The number of digits that are at exactly the same position in the **Guess** and the **Password**.
   (b) **Members:** The number of digits that exists in both **Guess** and the **Password** but at different positions.

   Some examples:

Guess	Password	Matches	Members	Desciption
`8761`	`5781`	1	2	`7` is in the same position in the **Guess** and the **Password**, hence the **Matches** is `1`. The digits `8` and `1` exists in both the **Guess** and the **Password**, therefore the **Agent** reports **Members** to be `2`
`9631`	`5093`	0	2	There are no identical digits at the same locations of the **Password** and the **Guess**, so the **Matches** is 0. The digits `9` and `3` exists in both the **Guess** and the **Password** but at different locations, so the **Members** is `2`
`7025`	`5481`	0	1	-
`7025`	`7043`	2	0	-
`9370`	`9370`	4	0	A perfect match, here the **CodeBreaker** wins the game.

4. The **CodeBreaker** inspects the report to improve the next **Guess**. Then the game continues from **Step 2**.
5. The **CodeBreaker** wins the game if the password can be uncovered in at most 8 trials, otherwise the **Agent** wins the game.

Questions

1. (20 points) Write a script that implements the **Agent**. The script should pick a valid random **Password** and ask the user for a guess, reporting the **Matches** and the **Members** after each guess. The game should end after 8 turns or when the user correctly guesses the password.

Sample run (User input is shown in **bold**, the **#$ python X.py** line is where the execution begins.)

```
#$ python agent.py

Enter guess> 1234
The Guess is 1234
Turn    Guess   Matches Members
1.      1234    0       3
Enter guess> 5678
The Guess is 5678
Turn    Guess   Matches Members
2.      5678    0       0
Enter guess> 2401
The Guess is 2401
Turn    Guess   Matches Members
3.      2401    2       2
Enter guess> 2410
The Guess is 2410
Turn    Guess   Matches Members
4.      2410    1       3
Enter guess> 2041
The Guess is 2041
You Win :)
The Password was: 2041
```

2. (40 points) Write a script that implements the **CodeBreaker**. In this case, the script will try to find the **Password** that the user picks. The algorithm for the **CodeBreaker** is more elaborate compared to the algorithm for the **Agent**. You can implement the following **CodeBreaker** algorithm:

1. Generate a list of all possible valid passwords, call this list `candidates`.
2. Pick a random item from the `candidates`, call it the **Guess**.
3. Get the number of **Matches** and **Members** from the user for the **Guess**.
4. Remove all items from the `candidates` that would not give the same **Matches** and **Members** for the **Guess** since these can not be the real **Password**.
5. Repeat steps 2 to 4 until the **Password** is cracked or the game is lost due to the turn limit.

Sample run (User input is shown in **bold**, the **#$ python X.py** line is where the execution begins.)

```
#$ python breaker.py

The Guess is 2146
Enter Report> 0 1
Remaining possibilities: 1260
Turn    Guess   Matches Members
1.      2146    0       1
The Guess is 9683
Enter Report> 1 1
Remaining possibilities: 198
Turn    Guess   Matches Members
2.      9683    1       1
The Guess is 1387
Enter Report> 1 1
Remaining possibilities: 34
Turn    Guess   Matches Members
3.      1387    1       1
The Guess is 9437
Enter Report> 4 0
Remaining possibilities: 0
I win!
```

3. (40 points) Write a script that implements both the **Agent** and the **CodeBreaker** together so that the computer plays the game by itself.

Sample run (User input is shown in **bold**, the `#$ python X.py` line is where the execution begins.)

```
#$ python selfplay.py

The Guess is 7823
Report: 1 2
Remaining possibilities: 207
Turn    Guess   Matches Members
1.      7823    1       2
The Guess is 8673
Report: 1 1
Remaining possibilities: 38
Turn    Guess   Matches Members
2.      8673    1       1
The Guess is 8527
Report: 0 2
Remaining possibilities: 11
Turn    Guess   Matches Members
3.      8527    0       2
The Guess is 4283
Report: 4 0
Remaining possibilities: 0
The Breaker Wins
The Password was: 4283
```

Grading

- This assignment is worth 10 points of your overall grade.
- Your code will be 50% of your assignment grade.
- Remaining 50% of your grade will be given in the lab based on your real-time coding abilities on similar algorithms.

How to submit

1. Write each individual answer in a separate file: `agent.py`, `breaker.py`, and `selfplay.py`
2. Create a zip file containing all your answers and name this zip file as `CNG111-ASN2-ID-20141.zip` where `ID` is your COMPLETE (7-digit) student ID.
3. Upload your zip file via `http://dropitto.me/cbasaran`
4. You will need to enter the password: `Metu_NCC_Fall_2014` to submit your assignment.
5. The deadline is final.
6. Plagiarism policy will be strictly enforced.