HW1

1. 1) García-Sánchez, P., Tonda, A., Mora, A. M., Squillero, G., & Merelo, J. J. (2018). Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone. Knowledge-Based Systems,153, 133-146. doi:10.1016/j.knosys.2018.04.030

   2) Fuller, M. Y., Mukhopadhyay, S., & Gardner, J. M. (2016). Using the Periscope Live Video-Streaming Application for Global Pathology Education: A Brief Introduction. Archives of Pathology & Laboratory Medicine,140(11), 1273-1280. doi:10.5858/arpa.2016-0268-sa

   3) Nagao, M., Tsujii, J., Mitamura, K., Hirakawa, H., & Kume, M. (1980). A machine translation system from Japanese into English. Proceedings of the 8th Conference on Computational Linguistics -. doi:10.3115/990174.990250

   4) Yang, L., Liu, J., Chen, C., & Chen, Y. (2016). Automatically proving mathematical theorems with evolutionary algorithms and proof assistants. 2016 IEEE Congress on Evolutionary Computation (CEC). doi:10.1109/cec.2016.7744352

2. 1) Performance: take into account the chosen card of hero with its specific kind of cards, build his/her deck before the game from a big pool of available cards, including both neutral and hero-specific cards

   Environment: the pool of available cards, player, computer

   Actuators: playtesting to check new card sets for possible design flaws, exploring the space of possible decks using an Evolutionary Algorithm

   Sensors: card sensor to learn which cards are in hand and which card in opponent's hand

   2) Performance: sharing knowledge, expertise, and research with potentially large audience without the barriers associated with traditional video equipment

   Environment: live video-streaming smart phone application(app) and the stability of the network

   Actuators: using the app for pathology education and scanning live videos

   Sensors: Recorded videos

   3) Performance: translating Japanese into English accurately and translation rate

   Environment: Japanese analysis, transfer, English synthesis components

   Actuators: using lexicon-based procedures and lexical descriptions, translate the Japanese video into English text

   Sensors: Stereotyped or semi-stereotyped expressions and sound sensor

   4) Performance: proving theorems based on intelligent behavior, generate or search for formal proofs directly in the realm of computation and the correctness of the proof of the theorems

   Environment: evolutionary algorithms, program generator, proof assistants, proof verifier

   Actuators: transformation from logic to computation, generate proofs directly

   Environment:

   Sensors: formula chart

3. 1) The state representation could be an array, so we can remove or add data in this data structure. Also the size of array can be changed.

   2) The first example given at Wikipedia, the initial state is {75, 50, 2, 3, 8, 7}

   3) CountDown(init):

```
CountDown(init):
    PQ=[init,h(init)]
    while not done:
        State=PQ.pop()
        If  is.goal(state):
        for  Si in successor(state):
            If  Si not in parents:
                PQ.insert(Si,h(Si))
```

   4) Explanations: My successor begins with choosing two elements from the array created above randomly and selecting one of the four basic operations. If the result is negative, it is not legal one. After this operation, we get a result that replaces the two numbers involved in the operation, as shown in example 1, the initial state is {75, 50, 2, 3, 8, 7}, the operation is

75+50=125, and the result is {125, 2, 3, 8, 7}. As this demonstrates, 75 and 50 have been replaced by 125. When there is only one number left, I will use a dictionary to store this final score and the numbers related to this number.

5) The branching factor would be $4*C_{n-m}^{2}$, we have 4 operations and we should choose 2 of the n-m numbers and m represents the current searching depth and n represents the initial states in the array.

6) The maximum search depth would be m-2

7) The answers will not change, because the initial state does not change and there are still 4 basic operations. Since we have only six cards, the search depth is fixed.

4. Path of maze1: [1, 2, 4, 8, 16, 13, 10, 20]

   Path of maze2: [1, 2, 4, 8, 5, 10, 20]

   Path of maze3: [1, 2, 4, 8, 5, 2, 4, 8, 5, 10, 7, 14, 28, 56, 53, 50]