Computational Problem Solving CSCI-603
Rescue Mission Lab 6

1 Introduction

Dutch and his team of elite soldiers are tasked by the CIA to rescue a group of officials being held hostage in an enemy base deep in a Central American jungle. The base is protected by a heavily armed force of insurgent guerrillas. Intelligence shows that the hostages are being held in a narrow cave inside the base with only one entrance and no exit - lined up from the front to back in standing position and unable to reorder themselves.

The rescue mission works as follows. One by one a soldier will leave their bunker and enter the nearby enemy base. Upon entering an alarm is activated which causes the first guerrilla to leave their line and investigate. The guerrilla engages the soldier and one of two scenarios play out:

1. The soldier is victorious over the guerrilla and the guerrilla is vanquished. In this case, the soldier rescues the one hostage at the head of the cave and escorts them to a nearby chopper. Next, the soldier returns to the back of the bunker to await their next rescue attempt.
2. The guerrilla is victorious and the soldier is vanquished. In this case, the one hostage attempting to be rescued adds themselves back to the head of the cave (effectively pushing the other hostages further into it) and the guerrilla returns to the end of the line to await their next encounter.

The chopper can hold up to 6 passengers aligned in a single columns of seats. There is only one door to the chopper that is accessible by the passengers. When they enter the chopper, the occupy the seat closest to the door and any existing passengers move one seat down. In order to preserve fuel, the chopper will only fly the passengers away to safety if the chopper is full, returning immediately to pick up the next set of passengers. This will continue until the last group of people have been rescued.

Unbenownst to the soldiers, an undetected alien spaceship has crashed into the same jungle area. One lifeform survived the crash - a ruthless creature known as the Predator. The Predator is the ultimate fighting machine. It is a beast who knows no mercy and hunts for game using its superior innate alien camoflauge and thermal tracking abilities.

In this homework, your will implement a simulation for the rescue mission in which each soldier will try to rescue a hostage and take them to the chopper. During the mission, a soldier will fight againts a member of the guerrilla and the predator.

Who, if anyone, will ”get to the choppa!”?

2 Implementation

2.1 Provided Files

You have been provided with the following:

1. The `output` folder contains sample runs. The numbers on the file names (e.g., 0 1 1 1) are the values of the command-line arguments used when running the program (see Main Program section).
2. The `doc` folder contains the documentation of the classes you will write.

2.2 Design

You have been provided with the documentation of the classes you will write for this lab. Take some time to familarize yourself with the design of the rescue mission simulation. Make sure you understand the roles and responsabilities of each class and how they interact with each other.

Notice that the documentation, exclude the private state and private behavior of each class. You will need to design that on your own.

2.3 Main Program

The main program should be named `rescue_mission.py`. This program will run the entire simulation. The program expects four integers as command-line arguments:

```
$ python rescue_mission #_seed #_hostages #_soldiers #_guerrillas
```

`#_seed` is the seed for the random number generator to simulate the die rolling.
`#_hostages` is the number of initial hostages in the simulation.
`#_soldiers` is the number of soldiers at the bunker.
`#_guerrillas` is the number of guerrillas in the enemy base.

If the arguments are present, they are guaranteed to be valid integer values. If not present, you should print an usage message an exit:

```
Usage: python rescue_mission #_seed #_hostages #_soldiers #_guerrillas
```

2.4 Implementation Details

During the simulation, a soldier will fight against members of the guerrilla and the predator. To determine the winner of every fight, the program will roll a dice, that is, it will generate a random number between 1 and 100 (both inclusive). To do so, use the `randint` function from the `random` module.

1. A guerrilla has a chance of **20 points** to defeat a soldier. If the die roll is greater than the chance to defeat the soldier, the soldier wins.
2. The predator has a chance of **75 points** to defeat a soldier. If the die roll is greater than the chance to defeat the soldier, the soldier wins.
3. If the predator defeats the soldier, then the hostage will fight against the predator. The predator has a chance of **50 points** to defect the hostage. If the die roll is greater than the chance to defeat the hostage, the hostage wins.

There is only one predator during the entire simulation. Notice that while the predator may be temporarily defeated by a soldier or hostage, it can never be truly defeated. The predator is around and ready to battle every soldier and hostage that comes out of the enemy base.

2.5 Seeding a Random Number Generator

Inside your program, import the `random` module, and call the function `random.seed(int_seed)`, where int_seed is the input integer seed value. Call this function only once, and do it before any other random number function calls in your program. This will cause the sequence of random numbers generated to be identical every time you run your program with the same int_seed value. This is helpful for debugging your program and checking to see if you get the same output as the sample runs provided.

2.6 Constraints

You may use the `Queue` and `Stack` data structures provided to you in lecture. You may not use any other Python data structures or advanced Python features not discussed in lecture.

3 Grading

Your grade will be determined as follows:

- 20%: results of the problem-solving
- 15% Design: The solution is object-oriented and breaks the problem down between various classes and methods.
- 55%: Functionality
- 10%: Code Style and Documentation

4 Submission

Create a ZIP file named `lab6.zip` that contains all your source code. Submit the ZIP file to the MyCourses assignment before the due date (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this lab).