# GEEN165 Game of Nim

Write a program to play the simple two player game of Nim. Nim starts with a pile of stones. We will assume there are 15 stones in the pile at the beginning. Players alternate turns taking 1, 2 or 3 stones from the pile. The player that takes the last stone from the pile loses. Your program should allow the user to play against the computer. Your computer program should win all the time!

When your program starts, it should use an Artificial Intelligence (AI) technique to learn how to play the game before playing the user. Your program can simulate playing many games to learn the proper number of stones to pick based on how many stones remain in the pile. When the computer simulates a game, it randomly chooses how many stones to take each time. At the end of the game, if it wins it increases the probability that it will pick that many stones next time. If it loses, it decreases the probability that it will pick that many stones in that situation next time.

Imagine the computer has 15 urns each containing 500 balls labeled 1, 500 balls labeled 2 and 500 balls labeled 3, a total of 1500 balls in each urn. When the computer needs to decide how many stones to remove when there are **`n`** stones in the pile, it picks a ball from the **`n`**th urn. Based on the number on the ball, the computer removes that many stones from the pile. The computer keeps the ball next to the urn. If the computer loses the game, it discards all of the stones next to the urns. If the computer wins, it puts the balls back in their associated urn and adds an extra ball with the same label to that urn. As the computer plays, the urns will fill with balls whose labels are likely to win while there will be fewer balls with labels likely to lose. Thus the computer is more likely to pick a ball with a winning number.

We can program the urns using a 15 by 3 array representing the 15 urns and the number of balls labeled 1, 2 and 3. Initially, each element of the array should be set to 500. When the computer randomly selects how many stones to remove, the selection is based on the relative number of balls in the urn or the counts in each of the three elements of the array for that number of stones. The following method returns 1, 2 or 3 randomly chosen based on the relative number of balls in the urn. The parameter is the 3 element array for the current number of stones on the pile. Assume `rand` is an object of the java.util.Random class.

```java
static int probPick(int[] odds) {
   int sum = odds[0] + odds[1] + odds[2];   // sum of all probabilities
   int randnum = rand.nextInt( sum );       // random number 0 to sum-1
   if (odds[0] >= randnum) {
      return 1;
   }
   else if (odds[0]+odds[1] >= randnum) {
      return 2;
   }
   else {
      return 3;
   }
}
```

When the program is playing the human or simulating a game, it calls the probPick method to decide how many stones to remove. The program needs another array of 15 integers to remember how many stones were removed. The array should be initialized to zero. The array element for that many stones in the pile is set to the number of stones removed in that game. At the end of a simulated game, the program should check each element of the array. If the value is not zero, it should increment or decrement the 15 by 3 array for that many stones removed depending on whether the computer won or lost the game.

We can make a simplification of the selection process when there is between 2 and 4 stones left on the pile. The program should always remove all but one stone from the pile assuring a win.

During the simulation the computer plays against itself. Consider two imaginary players A and B. Both imaginary players call probPick to decide how many stones to remove. Only player A needs to keep track of the stones selected and update the urns. After the two imaginary players simulate 100,000 games, the computer will become very good at winning. It should then play against the user.

**Sample output**

```
15 stones, pick 1-3 >3
9 stones, pick 1-3 >2
5 stones, pick 1-3 >1
You lose
```