GA Phase 2

Dear Students,

Below are the requirements for phase 2 of the project. As mentioned, this phase is about random solution generation. For this phase 2 (**Random**) and phase 3 (**GA**) to come, you are to use the same set of 3 inputs (**small, medium, large sized**) that you used in Phase 1 (**brute force**). You will be testing two random number generators:

1. Math.random() gives a random double from 0.0 (inclusive) to 1.0 (exclusive). See [**https://mkyong.com/java/java-generate-random-integers-in-a-range/**](https://mkyong.com/java/java-generate-random-integers-in-a-range/)
2. to generalize to numbers in a range other than 0..1

**b)** a generator based on the Blum-Blum-Shub (BBS) method. See **https://www.commonlounge.com/discussion/481152258acb4003a5903d2fc1bc425f** for a detailed description of the BBS method.

I will summarize the BBS algorithm now. This second generator has 3 parameters P, Q and X0. X0 is the "seed", the first number of the sequence as an input to get other numbers in the sequence (ie X0 is not used by your code; it's only purpose is to start things off.) For example, Let X0 = 127.

Set N = P\*Q. As an example let P = 67 and Q = 79 so that N = 67 × 79 = 5293. Then, the sequence will be generated iteratively by the following "iterative" formula:

Xi+1 = (Xi)^2 % N where ^ is raised to the power and % is the mod function. **NOTE:** that the Xi are ***only*** integers in the iterative generation of the sequence; **HOWEVER**, when ***using*** them, we are going to use it slightly differently since we are generating numbers between 0 and 1. You will take the integer generated and put a decimal point on the left. So, if Xi+1 = 250, then your code will output **0.250**; if Xi+1 = 4277, then your code will output **0.4277** and finally, if Xi+1 = 121, then your code will output **0.121** (Note: The above is a slightly simplified version of BBS.)

**IMPORTANT REMINDER:** for the purposes of ***generating*** the sequence, you only use the integers ***as is***, but the way you will ***utilize*** these numbers is to generate a number between 0 and 1 where the decimal points are the digits of the integer as explained above. The numbers of the ***sequence generated*** discussed above (250, 4277, 121) would actually be generated if X0 = 127. Namely, 127^2 (mod 5293) is 250; 250^2 (mod 5293) is 4277; and, 4277^2 (mod 5293) is 121. Thus, the ***sequence generated*** will be 250; 4277; 121; however, you will ***interpret*** these numbers as 0.250; 0.4277; 0.121 in order to accommodate a real number between 0 and 1.

Now, for each of the sets of input data you have from phase 1, you should generate at least 10000 random "solutions" to your problem, with a population of at least 1000 and ran for a minimum of 10 generations. You will need to define a fitness function for your problem whose maximum value would be from a brute-force solution and then keep track of the maximum fitness (and the solution that obtained it) and identify the minimum and average fitnesses for the all the guesses (random solutions) you generated. You will need to do this for both Math.random() based generator and BBS based generators separately (10000 guessed solutions for each) and for each of the input data sets, output the best random solution obtained by Math.random(), the best random solution obtained by BBS and their fitnesses. Please also include a brute force solution from phase 1 for each of these input data sets and compute the fitness of those as well. Sincerely, Professor