General Instructions:

- The submission due date of this assignment is **Saturday (19th March 2022) midnight** (before **12:00 am**)
- Write a report (i.e. in a word file) that illustrates your main solution steps including the best fitness values and the average values, plotted over generations (with and without elitism)
- Zip your code and the report in a file entitled [**YourName_YourID_AssignmentNumber**], submissions will be made on Blackboard.
- This assignment should be delivered and discussed INDIVIDUALLY

Requirements:

Write a Python code that implements the following:

1. Generate 20 chromosomes, each of which has a length of 5 binary digits (either 0 or 1).

2. Evaluate the fitness of each individual by counting number of ones in each chromosome (as demonstrated in the below table) and compare it to a target chromosomes of all ones (i.e. 11111)

Table-1: Fitness of a sample of chromosomes

Chromosome	Fitness
01110	3
01100	2
00000	0
10111	4

3. Write a function that selects the best fitted individuals for the next population using "**Roulette Wheel selection**", by taking the fitness of each individual and generates the probabilities of it. The probability of being selected is proportional to the relative fitness of the individual, that is calculated using the following formula:

$$p_i = \frac{f(x_i)}{\sum_{j=1}^{sum\_pop} f(x_j)}$$

Hence, the expected output of the individuals in Table-1 is [*0.333, 0.222, 0.444*]

4. Using the output of the function in step-3, implement another function that calculates the cumulative probabilities of each individual such they collectively sum to 1

Hence, the expected output of the individuals in Table-1 is [*0.333, 0.5556, 1*].
The selection of the best individuals will be based on their location in the cumulative probabilities output.

5. Write a function that implements "**one-point crossover**" that takes (two parents and a crossover probability pCross = 0.6) then returns a chromosome represents their offspring.

6. Write another function that implements a "**Bit-flip mutation**". It takes (a chromosome represents the individual, a mutation probability pMut = 0.05) and returns a mutated version of that individual.

7. Finally, let's put all together, write a function that takes (the population size = 20, number of generations = 100, chromosome length, probability of crossover and probability of mutation) and generates the final population, a vector containing the history of the highest fitness in the population at each generation 'best_hist' and the average fitness value as well.

8. Apply "**Elitism**" of size 2 to keep the best two individuals at each iteration.

9. Run your code different times (i.e. 10 runs) and change the random seed (Figure out how the results of the runs differ).

BEST OF LUCK!