# **ARTIFICIAL INTELLIGENCE**

**Assignment 2**

# **Genetic Algorithm**

Report

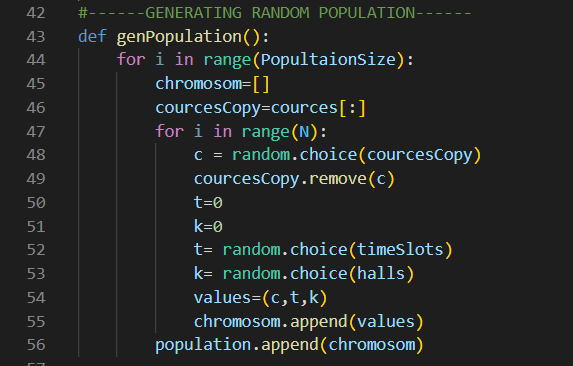
# **Harris Aamir**

# **20i-0943**

# **SE-S**

# Requirement 1:

The code contains multiple variables and list with description mentioned in comments. The user is asked for inputs for total number of cources, name of every course, exam duration of each course, number of exam halls, total time available for all halls, number of slots in a hall and number of common students between the cources. This data is then used to generate random population. The population consists of chromosomes with each chromosome representing a solution. The chromosome contains a schedule of each course containing the course name, slot and hall. The following code snippet shows the population generation function.



Text

Description automatically generatedThe following image shows some sample chromosomes. The first index of tuple is course name, second is time slot and third is hall.

# Requirement 2:

Text

Description automatically generated The fitness function is written to quantify the randomly generated solutions. The fitness function add a penalty based on the given constraints. If 2 cources are scheduled on same timeslot and in same hall it would add 1000 to the fitness value. Similarly, if the solution have a missing course fitness value increase by 4000. The solution is also checked for the exceeding hours. The algorithm calculates the total hours for each hall and checks if it exceeds the max time input by the user. For every extra hour fitness value increase by 10. Lastly the constraint of common students is checked where for every common student fitness increase by 100. The following code snippet shows the fitness function.

# Requirement 3

The algorithm runs for 100 generations and in each generation population have 80% of chromosomes from crossover , 10% from mutation and 10% from the best of last population.

The main function below first generates the random population and then saves the index and fitness of each solution in a list called index\_fitness. The next generation function generates the next generation population and returns back to main. After 100 generations the best solution index and its fitness is placed on the top of index\_fitness list. The best solution is printed.



The evaluate function and nextgen function are following

Text

Description automatically generated

Text

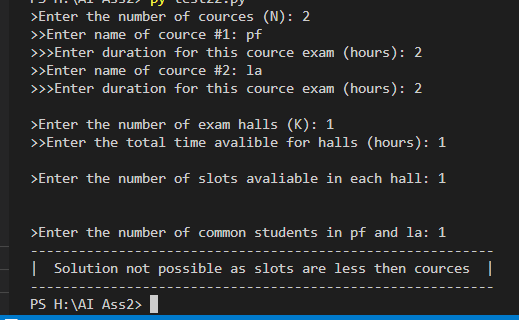
Description automatically generatedText

Description automatically generatedThe crossover and mutation functions are below

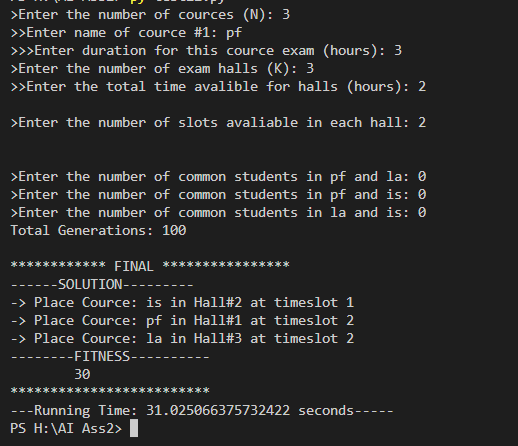
# Requirement 4:

The algorithm is tested upon different values

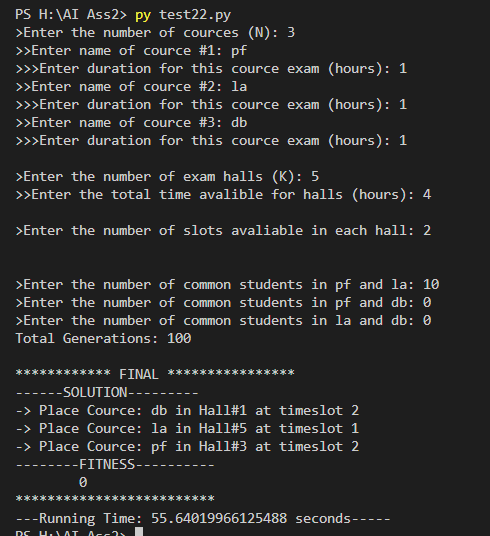
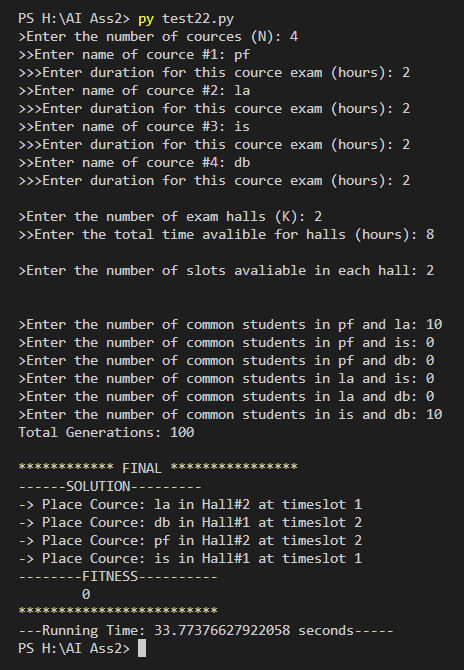
1. If the cources are less then possible slots, the program shows a message and exists



1. If the cources time duration is more then total time for hall



1. If there are common students among the cources. The algorithm places the cources at time which have the lowest common students. (in figure 2, pf and la have common student thus not at same time similarly is and db have common students which are also not in same time)



Ad

**Advantages of Genetic Algorithm:**

* Can find solution globally as the solution is found from a randomly generated population
* Can deal with constraints, the best solution is found on bases of fitness value that is calculated against the constraints
* They do not get stuck in local maxima as with crossover and mutation, more possible solutions are introduced into population
* Unlike hill climbing GA does not require information about the neighbors
* Easier to implement as only requires chromosomes formation, evaluation and deletion

**Disadvantages of Genetic Algorithm:**

* Slow as every generation generates new chromosomes and evaluated thus require more execution time and space
* Some solutions may not be written in chromosomes
* Depended upon different variables such as population size, selection method evaluation

🡸THE END🡺