Faziha Ikhlaq i20-0473

AI Assignment#2

<u>Report:</u>

This code implements a genetic algorithm to solve a course scheduling problem. The problem has a set of constraints, which are defined as follows:

- There are a number of courses that need to be scheduled.
- There are a limited number of exam halls available for the courses.
- Each course must be scheduled at a particular time slot in one of the exam halls.
- Each exam hall has a maximum number of hours available for scheduling.
- There are some conflicts between certain pairs of courses, which add to the fitness score of a solution.

The ***genetic algorithm*** works as follows:

A population of solutions is created, where each solution represents a valid schedule for all the courses.

The fitness of each solution is calculated, based on the constraints and conflicts of the problem. The fitness function takes a solution as input and returns a fitness value.

The algorithm selects the fittest individuals from the population to be parents of the next generation. Tournament selection is used, where a random subset of the population is selected and the fittest individual is chosen as a parent.

Crossover is applied to the parents to create offspring. This is done by selecting a random point in the solution and exchanging the sub-solutions after that point between the parents.

Mutation is applied to the offspring, which randomly modifies small portion of the solution. In this case, mutation involves randomly selecting a course and assigning it to a new exam hall and time slot.

The new generation is created from the parents and offspring, and the process is repeated for a fixed number of generations.

The best solution is chosen from the final population and returned.

- In my code, the **create_solution** function generates a random solution, and the **calc_fitness** function calculates the fitness of a solution. The **gen_population** function generates a population of solutions. The **tournament_select** function implements tournament selection, and the crossover and mutate functions implement crossover and mutation, respectively. Finally, the **run_algo** function puts everything together and runs the genetic algorithm for a given set of parameters.