MileStone 2

Pseudo Code

University Of Engineering And Technology

Submitted To: Sir Samyan Wahla

Project ID: CS311S20PIDG34

1. **Begin Genetic Algorithm**

//Tournament Selection uses in order to select two schedule to do crossover on

2. **select_tournamnet_population**(pop)

```
tournament_pop=population(0)

i=0

while i<TOURNAMENT_SELECTION_SIZE

tournamnet_pop.get_schedules().append(pop.get_schedules())[random(0,POPULATI
ON_SIZE)

i++

return tournamemt_pop
```

//It is iterative until it reaches to a fittest solution

3. **//Generates the Population of N randomly generated individuals**
4. **FUNCTION _crossover_schedule**(schedule1,schedule2)

```
crossoverSchedule=Schedule().initialize()

for i in range 0 to len(crossoverSchedule.get_classes()))

//Depending on random value we pickup classes from either schedule 1 and 2

   IF (rnd.random()>0.5)

    crossoverSchedule.get_classes()[i]=schedule1.get_classes()[i]

   ELSE:

    crossoverSchedule.get_classes()[i]=schedule2.get_classes()[i]

RETURN crossoverSchedule
```

//Does mutation on rest of the schedules

5. **crossover_population**(pop)

```
crossover_pop=Population(0)
```

```
   //before doing crossover population here we will append
NUMB_OF_ELITE_SCHEDULES without changing them (we only have one
(NUMB_OF_ELITE_SCHEDULES) with the highest fitness.)

    for i=0 to NUMB_OF_ELITE_SCHEDULES

       crossover_pop.get_schedules().append(pop.get_schedule()[i]

       i=NUMB_OF_ELITE_SCHEDULES

//For remaining schedule we will do tournament selection and pickup the fittest
   schedule

      while i<POPULATION_SIZE

        schedule1=select_tournament_population(pop).get_schedules()[0]

        schedule2=select_tournament_population(pop).get_schedules()[0]

     i++

//by coding crossover population on schedule 1 and schedule 2 it returns the
   fittest schedule

   return crossover_pop
```

//This method mutate the passes in schedule before returning it

6. FUNCTION _mutate_schedule(mutateSchedule)

```
  //Schedule() function returns the classes of a schedule by handling conflicts
and fitness (classes include department name, course number, room number,
teacher id and class time.)

    schedule=schedule().initialize()

    for i in range  0 to len(mutateSchedule.get_Classes()))

 //IF random number is smaller or equal to the mutation Rate (this mutation would
   often happen that's why we set mutation rate = 0.1)then we do mutation

       mutateSchedule.get_Classes()[i]=schedule.get_Classes()[i]

     RETURN mutateSchedule
```

//It end up calling crossover population and then calls mutate

```
7. evolve(population)
    return mutatePopulation(crossover_population(population))
```

//here we evolve the population from one generation to next until we get to the population and fittest schedule has zero conflicts

```
8. while population.get_schedules()[0].get_fitness()!= 1.0)
    generationnumber += 1
    output Generation # + str(generationNumber)
    population =geneticAlgorithm.evolve(population)
    population.get_schedules.sort(get_fitness , reverse=true)
    population.get_schedules()
```

```
9. //after that it will  display the fittest schedule
```

//Does mutation on rest of the schedules

```
Procedure _mutate_population(population)
  for i = NUMB_OF_ELITE_SCHEDULE  to POPULATION_SIZE
      mutate_schedule(population.get_schedules()[i])
      RETURN population
End Procedure
```