TASK 1

Langton’s Ants

Langton’s Ants is a two-dimensional Turing machine simulation where an ant moves according to simple rules, flipping the color of squares and changing direction as it traverses the grid. Over time, its behavior leads to emergent complex patterns, including chaotic movement and structured path.

Squares on a grid are colored either white or black. We arbitrarily identify **two** squares as the "ants." An ant can travel in any of the four cardinal directions at each step. The ants have a peculiar characteristic - they release **pheromones**, which help them keep track of their path. Each ant recognizes its pheromone; however, sometimes, due to the similarity of the scents released by other ants, it may get confused and behave weirdly. These are the specifications of the simulation you need to code:

1. Basic Movement Rules:
   - At a white square,
     - they turn **90° clockwise**
     - flip the color of the square
     - release a pheromone (for example, as A for one ant and B for the other)
     - move forward one unit
   - At a black square,
     - they turn **90° counter-clockwise**
     - flip the color of the square
     - release a pheromone
     - move forward one unit

2. Pheromone Influence:
   a. Self-pheromone Recognition

An ant, on recognizing its pheromone, understands that it has visited that square before and it does not apply the standard turning rule. Instead, it moves straight ahead with a *probability of P = 0.8*, with *only a 20% chance* of following the standard turning rule.

b. Cross-pheromone Recognition

If an ant encounters the pheromone of the other ant, the probabilities are reversed.

c. Pheromone Replacement

Once a new pheromone is released on any square, any existing pheromones at that square are removed.

d. Pheromone Decay

The influence of a released pheromone (i.e., *the probability it causes the ant to move straight*) decays linearly over the time it takes for an ant to move approximately five squares.

IMPORTANT POINTS TO REMEMBER:

1. Python is the preferred programming language for this task. Refrain from using any other language.
2. You may use the *Pygame* library for the simulation interface, though you are free to choose any other suitable library based on your preference.
3. Make sure the code is well-structured and readable.
4. Bonus points will be awarded for using *OOPs principles*.

DELIVERABLES:

1. A .py file of the solution code, well-commented with all the necessary instructions for running the code if any. Irrespective of what library you use, a simulation interface must be present, in which the game is running.

RESOURCES:

1. Langton's Ant
2. Langton's Ant Algorithm - Python and Pygame
3. PyGame documentation