Dmitri Stanchevici
Unit 4, Assignment 1

A1.1

// This program simulates an ecological interaction of three entities: predator, prey, and food. Predators move toward and eat prey, prey move toward and eat food, while food is stationary. The goal is to determine how many steps are needed for the predators to eat all the prey. The entities are depicted on GUI as dots.

Create class Entity with properties for location
    double x, y;

Create Main class
    Initialize three linked lists, one for each kind of entity
        predatorList, preyList, foodList

    Assign random locations to each entity and place all of the entities on screen (ecosystem).

    Initialize two remove lists
        preyRemove and foodRemove

    Specify the number of steps and distance that predators and prey move at every step.

    Specify the distance(s) at which the predator can eat prey and prey food.

    Assign random positions to each entity on each of the three lists.

    Create the ecosystem (that is start a new GUI window with the randomly positioned dots for the entities).

    For from 0 to number-of-steps, call three methods:
        move()
        eat()

    In move()
        In a double loop determine each predatory entity's closest prey entity.
        Move this predator entity toward this prey entity by the specified distance.

        In a double loop determine each prey entity's closest food entity.
        Move this prey entity toward this food entity by the specified distance.

In eat()

In a double loop, compare predator and prey distances

If distance between a predator entity and prey entity < determined distance

add prey entity to preyRemove

In a double loop, compare prey and food distances

If distance between a prey entity and food entity < determined distance

add food entity to foodRemove

In a double loop,

remove items found in preyRemove from preyList

Empty preyRemove.

In a double loop,

remove items found in foodRemove from foodList

Empty foodRemove.

Problem 5: Operating System

What Is An Operating System?

An operating system (OS) is a computer program that acts as an intermediary between applications and hardware. Modern computers--including servers, PCs, tablets, and smartphones--are often sold with an OS pre-installed on them. Today's best-known and most widely used OS families are MS Windows, macOS, Lynux, and Android. The absolute majority of users interact with computing devices through the GUI interface provided by an OS. Users also organize their work through the filing systems provided by operating systems. OSs also protect computers from viruses and unauthorized access. Unbeknownst to most users, the OS ensures a smooth and efficient interaction between hardware and programs by allocating and deallocating memory; scheduling processing times and thus enabling the performance of multiple tasks concurrently (or, more precisely, intermittently, thus creating a convincing illusion of concurrency); connecting I/O and networking devices to the main CPU and memory by means of device drivers; and ensuring memory and program security by giving each program a portion of memory and by preventing programs from spilling over into other programs' portions

of memory (https://www.youtube.com/watch?v=9GDX-IyZ_C8). Finally, as a layer of abstraction underlying applications and overlaying hardware, OSs standardize procedures for programming and operating diverse ranges of computing devices. For example, operating one Windows machine is very similar to operating another Windows machine, despite possible differences in the CPUs, memories, and peripheral devices. Such standardization means not only that users and programmers do not have to learn how to use each new piece of hardware, but also that programs can be easily reused on different computers.

Historically Important Operating Systems

The evolution of operating systems was driven by the need for more efficient ways to load programs into memory and to free the memory after the completion of the program. These tasks used to be performed by human computer operators feeding into special reading devices punched cards or punched tapes, created and delivered to the room housing the computer by programmers. Computers could read only one program (one batch of cards) at a time. Thus this process was slow and left the highly expensive computing unit idle for long periods of time. In the 1940s and 50s, computers gradually evolved to read several such batches in successions. An early program that enabled such multiprocessing was the **Atlas Supervisor,** created in 1962 especially for the powerful Atlas computer at the University of Manchester (https://www.youtube.com/watch?v=26QPDBe-NB8). Another challenge programmers faced was writing extra code for operating different models of peripheral devices, such as printers. In the 1950s and 60s, standard subroutines for such operations were created and widely distributed; they were precursors of the device drivers OSs use today (https://people.cs.rutgers.edu/~pxk/416/notes/01-intro.html).

Multics, developed in the late 1960s by MIT, General Electric, and Bell Labs, is considered an important achievement in the history of multifunctional operating systems. This OS provided multiprocessing, memory security, a sophisticated filing system, I/O interoperability, and other features that define today's operating systems. One drawback of this powerful program was its high resource intensity, barring it from use on smaller machines. To solve this problem, Dennis Ritchie and Ken Thompson, two of Multics developers at Bell Labs, created a leaner operating system called **Unix**. Because Unix had shed many of the features of Multics, it was not as reliable and it required rebooting the system more often than Multics did; however, Unix was also much simpler and cheaper than Multics, making it a popular choice in the 1970s and an

important model for modern OSs. In fact, today's **macOSs** and **Lynux OSs** are described as "Unix-like." They do not use the proprietary code of Unix, but their designs reflect Unix's functionality and architecture.

The 1980s saw the rise of personal (or home) computers, which required relatively simple and inexpensive OSs. One such OS was **MS-DOS.** This command-line based OS came with the computer on a disk that was just 160 kilobytes. MS-DOS, as well as earlier versions of GUI-based **MS Windows** lacked sophisticated memory security and reliability, so a program's error could easily crash the whole operating system. However, these OSs played an important role in democratizing personal computers. Since its early days MS Windows has made considerable progress, improving its security and usability. These improvements, plus clever marketing, have ensured that MS Windows has been for decades, and it still is, the most widely used OS around the world.