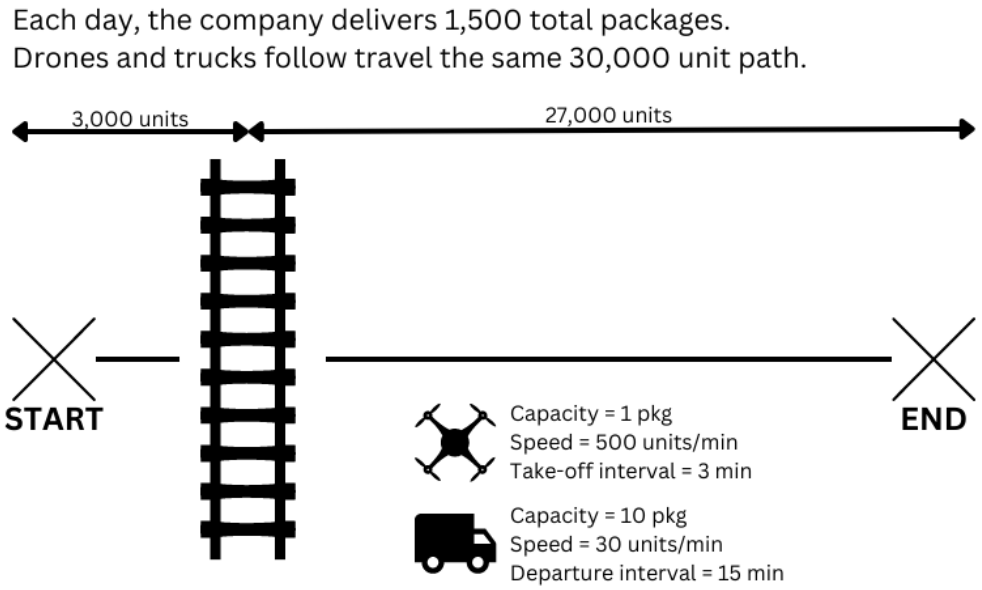
CS 1181 Project 3: Data Science (Package Delivery Simulation)

PURPOSE: This assignment explores simulation of a real-world situation which leads to meaningful actions to solve a problem. A very common use of computers is to run simulations to answer practical questions with no easy closed-form mathematical solution. A simulation is a program designed to model a real system of interest. By setting parameters and selecting an appropriate model, results are produced that model the behavior of the real system. If the model is a valid representation of the real system, then the simulation can even predict the future performance of the real system.

PROBLEM: For this project, you will be simulating a package delivery system involving drones and trucks. A delivery company is interested in decreasing the time it takes them to deliver packages. Currently, the company only uses trucks to deliver packages, but a railroad crossing has been causing costly delays. Thus, the company purchased a number of drones capable of delivering packages without the delays of the crossing. Now, the company wants to optimize their delivery system but is uncertain how many packages should be delivered by the new drones. Your goal is to determine what percent of packages should be delivered via drones (vs trucks) to result in the shortest delivery time of all packages.

The following constant information has been provided to you:

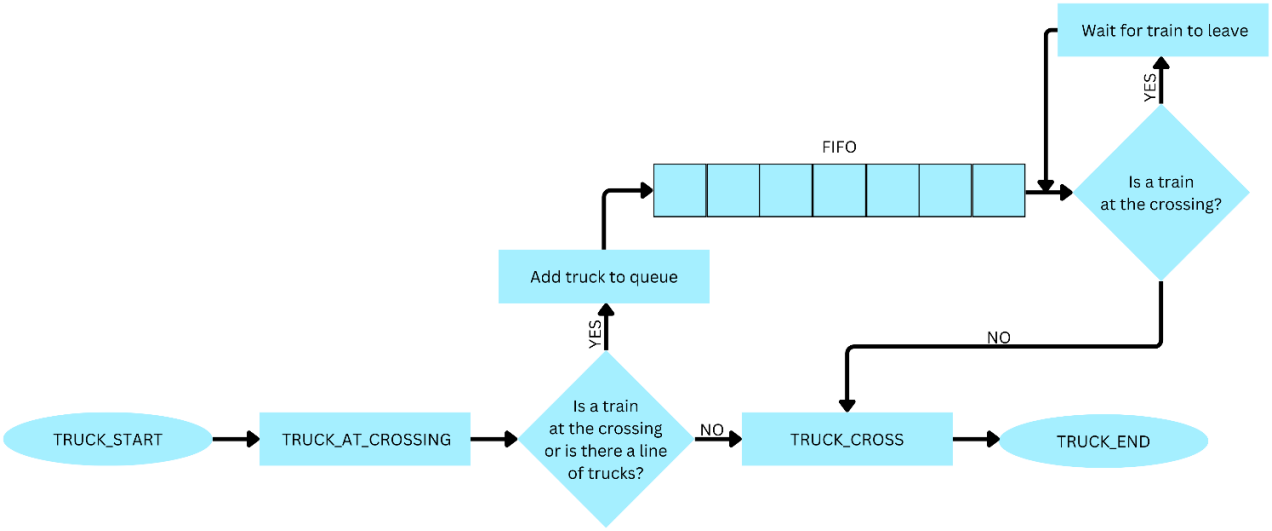


train_schedule.txt: This file contains a list of start times and durations that the train blocks the track in minutes. For example, the first line of the file reads "94 77". This means that a train begins blocking the crossing at 94 minutes and continues to block the crossing for 77 minutes. To find the time that the train is no longer blocking the crossing, you can add the duration to the start time (i.e. 94 + 77 = 171).

Number of Drones and Trucks: Create a variable called PERCENT_BY_DRONE that indicates what percent of the packages are delivered by drone. Using this value, you need to determine the number of drones and trucks that are needed. Remember that each drone can carry one package and each truck can carry ten packages. Therefore, if you need to deliver 100 packages, and PERCENT_BY_DRONE is .5 (50%), then you would need 50 drones (100 * .5 = 50) and five trucks ((100 * .5) / 10 = 5). If the number of packages delivered by truck does not divide by ten evenly, round up as an additional truck is necessary. To determine the total time to deliver all packages, the total times for the drones and trucks to complete their deliveries must also be determined individually.

Drone Calculations: You need to calculate the time it takes for one drone to complete its trip and the total time it takes for all drones to complete their trip. The first drone should take off at time 0.0. Keep in mind that multiple drones may be flying at the same time, so the total time is *not* the sum of each drone's trip time. Since the drones do not have any obstacles, algebra can be used to determine the total time.

Truck Calculations: Because the trucks can be interrupted by the train, ***discrete event simulation*** must be used to determine the truck calculations. In discrete event simulation, time is managed using a simulation clock that is advanced whenever an interesting event occurs. For example, if the current simulation time is 5.0 minutes and a truck begins its route at a time of 10.0 minutes, we mark the passage of time by advancing the simulation clock to 10.0 minutes. If the next event to occur is a truck reaching the crossing at 35.0 minutes, we advance the clock to 35.0. It is assumed that no interesting activities occur between events (i.e. nothing significant happens in the intervals (5.0, 10.0) or (10.0, 35.0), so the passage of time is recorded by moving the simulation clock forward to the time when the next important event occurs).



For this project there are several different events of interest, consisting of train events and truck events. Below is a brief description of each of the significant truck events. Keep in mind that all trucks travel at the exact same speed on a one-lane road, so under no circumstances may one truck pass another.

- TRUCK_START: The first truck starts at time 0.0. Due to the required offset between truck departures, the second truck will start at 15.0. Since the truck speed and distance to the tracks have been provided, a simple calculation can be used to determine when the truck will arrive at the crossing.

- TRUCK _AT_CROSSING: Here two outcomes may occur. First, if there is no train blocking the crossing, the truck may cross immediately and continue its journey. The other possibility is that the truck must wait. If a train is blocking the crossing, or if no train is present, but there is a line of trucks waiting to cross, then the truck must wait. Therefore, it is possible that a truck may have to wait even if the train is already gone. Each truck must also take one minute to start up from a stop and cross. For example, if a train leaves the crossing at time 35.0, then the first truck waiting will cross at 36.0, followed by the next truck at 37.0, and so on until all the trucks have crossed.

- TRUCK_CROSS: Once the truck crosses the tracks, algebra can be used to determine when the truck will complete its delivery.

- TRUCK_END: After the truck completes its journey, the time and any other statistics can be recorded.

In addition to truck events, you need to keep track of the significant train events: when a train begins to block the crossing and when it is no longer blocking the crossing.

Event Sorting/Other Considerations: To process the events, they should be stored in a data structure that allows them to be sorted by event time. Additionally, some sorting rules must be implemented in case multiple events occur at the same time. First, a train event should always take priority over a truck event. Second, a truck arriving at the crossing should be handled before a truck that is crossing at the same time. This ensures two trucks do not cross at the same time.

Via proper implementation of the discrete event simulation and sorting, your solution should demonstrate that no truck is able to cross the crossing when a train is present. Additionally, it should be clear that your solution properly queues trucks that cannot cross and dequeues them when the train is gone. This includes ensuring that a new truck arriving at the crossing does not cut to the front of a pre-existing line of trucks.

Real-Time Tracking: For each significant event, a statement including the time that the event occurs, the type of vehicle (truck or train), the id (if a truck), and a brief description of the event should be printed to the console in correct time order. For example, if the second truck is waiting at the crossing, "76.0: TRUCK #2 waits at crossing" could be printed to the console. Only the truck and train events need to be tracked because they use discrete event simulation. Below is a list of every event type that should be printed to the console:

- Truck beginning its journey
- Truck crossing the crossing
- Truck waiting at the crossing (if necessary)
- Truck completing its delivery
- Train blocking the crossing
- Train no longer blocking the crossing

Statistics: After the simulation completes and the drone calculations are performed, print the following to the console:

- Number of drones/trucks for the simulation
- Full train schedule
- Total trip time for each truck (time from truck departure to time it reaches destination)
- Average trip time for all trucks
- Total time for all trucks to deliver packages
- Trip time for one drone (same for every drone)
- Total time for all drones to deliver packages
- Total time for all packages to be delivered

(Remember that the total times for both the drones and the trucks are not the sum of all drone and truck trip times respectively because multiple drones and trucks are traveling simultaneously.)

End Goal: Once the simulation runs properly and the statistics are calculated correctly, run your program several times with different values for PERCENT_BY_DRONE. *In a comment at the top of your main file*, indicate what PERCENT_BY_DRONE produces the lowest total time for all packages to be delivered, rounding to the nearest percent.

GRADING

(75 pts) **Base Functionality**
- [10] "train_schedule.txt" is correctly read and parsed
- [5] The number of drones and trucks is calculated correctly given a certain PERCENT_BY_DRONE
- [10] Drone calculations are correct (individual drone trip, total drone trip time)
- [10] Truck calculations are correct (total trip time per truck, total and average trip time for all trucks)
- [15] Events are sorted properly, trucks properly queue, and dequeue when the train is gone
- [10] Real-time tracking information is printed to the console
- [10] Truck and drone statistics are printed to the console
- [5] Final best value for PERCENT_BY_DRONE is correct

(25 pts) **Style**
- [10] Object-oriented principles such as encapsulation, inheritance, and polymorphism, are used appropriately
- [10] The coding is ***meaningfully*** commented, and there is not commented out junk code
- [5] Standard coding conventions are followed, including variable names and indentation

NOTE: There is no valid excuse for turning in a program which does not compile. A program will receive no credit if it does not compile and execute.