Problem Set 3

Problem 1a)

The full code for this can be found HERE

We simulated a shared vehicle station where

- Vehicles arrived according to a poisson process with rate $\lambda = 6$
- There are three types of clients requesting bikes:
  - Class 1: members rate $\mu_1 = 3$, fee $K_1 = 0.5$, penalty $c_1 = 1.0$
  - Class 2: members rate $\mu_2 = 1$, fee $K_2 = 0.1$, penalty $c_2 = 0.25$
  - Class 1: members rate $\mu_3 = 4$, fee $K_3 = 1.25$, no penalty

The simulation runs for T = 120 time units start with X(0) = 10 initail vehicles.
We test the simulation with difference number of occurence with the results being

Occurences	Result
10	\$565.1
100	\$553.793
1000	\$552.23
10000	\$551.69
100000	\$551.71

```
double arrivalTime = randomExponentialNumberGenerator(1.0 / lambda);
double client1ArrivalTime = randomExponentialNumberGenerator(1.0 / client1Rate);
double client2ArrivalTime = randomExponentialNumberGenerator(1.0 / client2Rate);
double client3ArrivalTime = randomExponentialNumberGenerator(1.0 / client3Rate);

// more code
```

We use discrete event simulation tracking four event types: bike arrivals and three client requests. Interarrival times are generated using **randomExponentialNumberGenerator** with appropriate rates. At each step, we find the next event (minimum time among all events), advance the clock, update the state (occupancy, penalties, rides), and generate new event times. This continues until time ≥ T.

After time T, the membership revenue is calcualted with
$\text{Membership revenue} = (K_1\mu_1 + K_2\mu_2) * T$
$\text{Ride revenue} = (\text{client3Rides}) * K_3$
$\text{Penalties} = c_1(\text{client1Penalties}) + c_2(\text{client2Penalties})$

We use 10,000 replications as the estimate stabilizes, yielding an estimated net profit of \$551.69 since the accuracy of the model didn't show much improvment after this point.

Problem 1b

i)

If we merge all the poisson processes we get a superposition of independent Poisson processes with rates $\lambda_{\text{Total}} = \lambda + \mu_1 + \mu_2 + \mu_3 + \lambda = 6 + 3 + 1 + 4 = 14$

The total number of events in the distribution M would be

$M \sim \text{Poisson}(\lambda_{\text{Total}} * T) = \text{Poisson}(14 * 120) = \text{Poisson}(1680)$

Due to **Order Statistics**, given M events in $(0,\ \text{T}]$, the event times are distributed as the order statistics of M uniform random variables on $(0,\text{T}]$.

ii)

To generate M we can use the function in problem-set-2 to generate a random variable using the

$M \sim \text{Poisson}(\lambda_{\text{Total}} * T) = \text{Poisson}(1680)$

```
double generateM(double lambda, double timeInterval) {
  double randomNumber = randomFloatGenerator(0, 1);

  return transformationMethodPoisson(lambda * timeInterval, randomNumber);
}
```

iii)

The full code for this can be found HERE

Given that an event occurred in the merged process the probability it's of each type is:

- P(Arrival | event) = $\frac{\lambda}{\lambda_{\text{Total}}} = \frac{6}{14}$
- P(Class 1| event) = $\frac{\mu_1}{\lambda_{\text{Total}}} = \frac{3}{14}$
- P(Class 2| event) = $\frac{\mu_2}{\lambda_{\text{Total}}} = \frac{1}{14}$
- P(Class 3| event) = $\frac{\mu_3}{\lambda_{\text{Total}}} = \frac{4}{14}$

Then we can change our code to choose a random event based on these probabilties,

```
  double lambdaTotal = lambda + client1Rate + client2Rate + client3Rate;

  std::vector<double> eventWeights = {lambda, client1Rate, client2Rate,
client3Rate};
  std::vector<double> results = {};
  results.reserve(numberOfReplication);

  for (int i = 0; i < numberOfReplication; i++) {
    int M = generateM(lambdaTotal, timeInterval);

    double time = 0.0;
    int occupancy = initalOccupancy;

    int client3Rides = 0;
    int client1Penalties = 0;
    int client2Penalties = 0;

    for (double j = 0; j < M; j++) {
      int M = generateM(lambdaTotal, timeInterval);
```

Again we choose to use 1000 replications since the estimation did improve much after this point.