CSC412: Modeling and Simulation **Midterm Exam**

Student name: mishari albuhairi **id:** [redacted]

1. Define and discuss the meanings of the following simulation terms

a. Model: An abstraction of reality, representing physical or non-physical systems, phenomena, or processes

b. Environment: A virtual created setting that mimics real world systems or processes for simulation

c. System: A representation in simulations, encompassing selected elements and boundaries to depict a simplified version of reality.

e. Computer Simulation: The use of computer-based mathematical models to emulate the behavior or outcomes of real-world systems.

f. Continuous Model: A simulation approach using continuously changing variables, typically governed by differential equations.

g. Discrete Event Model: A simulation technique that models system changes as a sequence of distinct, individual events.

h. Model Accuracy: The extent to which a simulation model accurately reflects the real world scenario it is designed to represent.

i. Model Performance: The efficiency and effectiveness with which a simulation model operates and produces results.

j. Quality of a Random Number: An assessment of how well a random number generation process simulates true randomness in simulations.

2. Consider a random variable X which takes on values 0 and 1 with probability 0.4 and 0.6, respectively.

a. Determine the mean and variance of X.

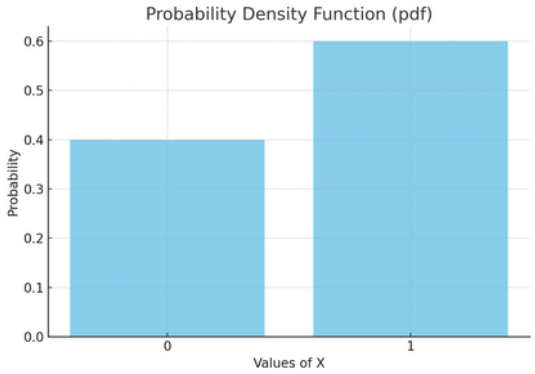
$E(X) = \mu = \sum xP(x) = 0\times0.4+1\times0.6 = \mathbf{0.6}$

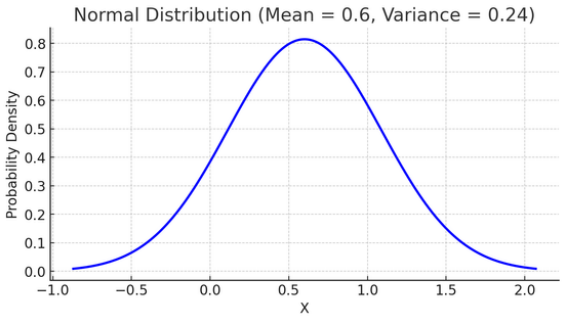
$\sigma 2 = \sum(x-\mu)2P(x) = (0-0.6)2\times0.4+(1-0.6)2\times0.6 = \mathbf{0.24}$

b. Plot the probability density function (pdf)

c. Plot the probability distribution function (PDF) of X.

X	P(X)
0	0.4
1	0.6

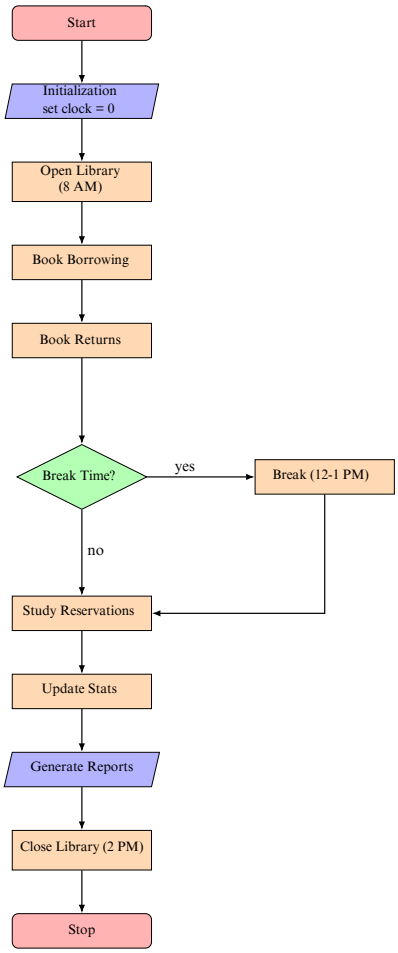




3. Sketch a flowchart for a discrete-event simulation model

I sketched a flowchart model simulation for king saud university library with taking into account:

–System state – variables to describe state
–Simulation clock – current value of simulated time
–Event list – times of future events (as needed)
–Statistical counters – to accumulate quantities for output
–Initialization routine – initialize model at time 0
–Timing routine – determine next event time, type; advance clock
–Event routines – carry out logic for each event type
–Library routines – utility routines to generate random variates, etc.
–Report generator – to summarize, report results at end
–Main program – ties routines together, executes them in right order



as we learned in lectures:

1. Initialization Routine: Sets initial conditions.
2. Main Program: Controls the simulation flow.
3. Timing Routine: Manages event scheduling.
4. Event Routine: Executes event-specific actions.
5. Library Routines: Supports with utility functions.
6. Report Generator: Summarizes simulation data.
7. Break Time Check: Pauses for scheduled breaks.

4. Write a program that models a single server queue with deterministic interarrival and service times. Give its output and discuss the results.

```
import random
import simpy

# Parameters
SIM_TIME = 50  # Simulation time in minutes
SERVICE_TIME = 5  # Average service time in minutes
INTERARRIVAL_RATE = 5  # Average time between arrivals in minutes

class SingleServerQueue:
    def __init__(self, env, service_time):
        self.server = simpy.Resource(env, capacity=1)
        self.service_time = service_time
        self.arrived = 0
        self.served = 0
        self.waiting_times = []
        self.customers_not_served = 0

    def service(self, customer):
        """Service process. It takes a random service time."""
        yield env.timeout(random.expovariate(1.0 / self.service_time))
        self.served += 1
        self.waiting_times.append(env.now - customer.arrival_time)

class Customer:
    def __init__(self, name, arrival_time):
        self.name = name
        self.arrival_time = arrival_time

def handle_customer(env, customer, queue):
    """Customer process. Each customer arrives,
requests the server, gets served and leaves."""
    arrival_time = env.now
    print(f"{arrival_time:.2f}: {customer.name} arrived")
    with queue.server.request() as request:
        yield request
        wait_time = env.now - arrival_time
        queue.waiting_times.append(wait_time)
        print(f"{env.now:.2f}: {customer.name} being served")
        yield env.process(queue.service(customer))
        print(f"{env.now:.2f}: {customer.name} finished being served")

def customer_arrivals(env, queue):
    """Generate new customers that arrive at the queue."""
    while True:
        arrival_time = env.now
        customer = Customer(f'Customer {queue.arrived + 1}', arrival_time)
        queue.arrived += 1
        env.process(handle_customer(env, customer, queue))
        yield env.timeout(random.expovariate(1.0 / INTERARRIVAL_RATE))

# Setup and start the simulation
env = simpy.Environment()
queue = SingleServerQueue(env, SERVICE_TIME)
env.process(customer_arrivals(env, queue))
env.run(until=SIM_TIME)

# Calculate statistics after the simulation
customers_not_served = queue.arrived - queue.served
mean_waiting_time = sum(queue.waiting_times) / len(queue.waiting_times) if queue.waiting_times else 0

print('--- Simulation Finished ---')
print(f'Customers arrived: {queue.arrived}')
print(f'Customers served: {queue.served}')
print(f'Mean waiting time: {mean_waiting_time:.2f} minutes')
print(f'Customers not served: {customers_not_served}')
```

```
0.00: Customer 1 arrived
0.00: Customer 1 being served
1.52: Customer 2 arrived
1.65: Customer 1 finished being served
1.65: Customer 2 being served
3.79: Customer 2 finished being served
9.51: Customer 3 arrived
9.51: Customer 3 being served
10.77: Customer 3 finished being served
13.49: Customer 4 arrived
13.49: Customer 4 being served
13.67: Customer 4 finished being served
27.30: Customer 5 arrived
27.30: Customer 5 being served
27.55: Customer 6 arrived
29.40: Customer 5 finished being served
29.40: Customer 6 being served
33.38: Customer 7 arrived
39.42: Customer 6 finished being served
39.42: Customer 7 being served
41.44: Customer 7 finished being served
45.43: Customer 8 arrived
45.43: Customer 8 being served
45.96: Customer 9 arrived
46.80: Customer 8 finished being served
46.80: Customer 9 being served
48.65: Customer 10 arrived
--- Simulation Finished ---
Customers arrived: 10
Customers served: 8
Mean waiting time: 2.21 minutes
Customers not served: 2
```

after pluging the code with

- SIM_TIME = 50 # Simulation time in seconds
- SERVICE_TIME = 5 # Average service time in sec
- INTERARRIVAL_RATE = 5 # Average time between arrivals in sec

and generating customers randomly using python library **RANDOM**

we get these results shown

(results will be different every run since customers are randomly generated)

--- Simulation Finished ---
Customers arrived: 10
Customers served: 8
Mean waiting time: 2.21 minutes
- **Customers not served: 2**

5. Give your expertise about how you could model and simulate traffic in Riyadh (only the main roads around KSU. Discuss the model performance and accuracy concerns. Simulate the model and give its output during rush hours.

1. mapping out the main roads:
- **Abdullah ibn saad Road**
- **turki 1st Road**
- **king khalid Road**
- **king abdullah Road**

2. Traffic Flow Data (assumption):

SIM_TIME = 180 # Total simulation time in minutes (3 hours to represent rush hour)
SIGNAL_CYCLE = 45 # Signal cycle time in seconds (green light duration)
CARS_PER_GREEN = 30 # Number of cars that can pass during a green light
INTERARRIVAL_RATE = 0.5 # Average time between arrivals in minutes (dense traffic, car every 6 seconds)
ROAD_CAPACITY = 50 # Capacity of cars on each road
ROADS = ['Imam Road', 'Turki 1st', 'King Khalid Road', 'Abdullah ibn saad Road']
Names of the roads around the university and each road has a entry and exit

- king khalid road doesnt have a signal light so that cant calculate the queue in this simple program

3. **Data Accuracy**: Make sure the information about traffic is current and correct. Use data from traffic sensors or studies.

4. **Changing Traffic**: Traffic can change quickly. The model needs to handle different random situations like accidents, closed roads, or bad weather. on complex models

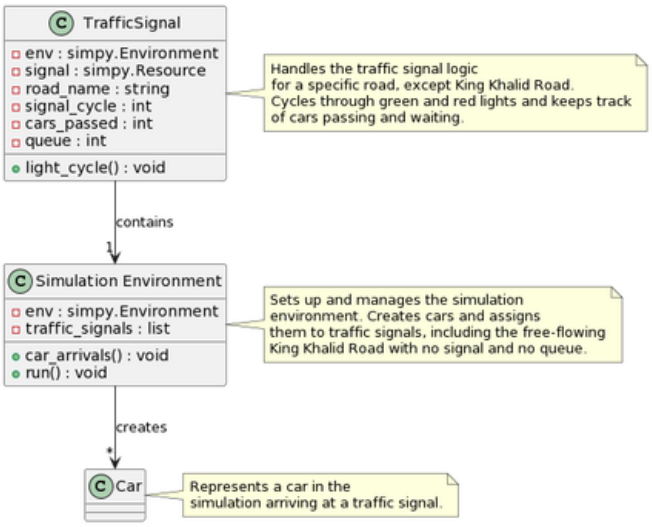
5. **Testing the Model**: Use real data to fine-tune the model. Check that the model's results try to match real-world traffic to make sure it's reliable.

6. **Complexity vs. Simplicity**: Traffic models can be complex and require a lot of computing power. It's important to find a good mix of being detailed enough to be accurate but simple enough to manage efficiently.

7. **Dynamic Variability**: This means that traffic can change a lot and quickly. The model you use should be flexible enough to handle different kinds of traffic situations. **Example from Code**:, dynamic variability can be illustrated by how the **car** function handles cars at traffic signals. Imagine a scenario where a road is temporarily closed due to an accident. In the model, this could be simulated by increasing the **signal_cycle** time for the affected road, representing the delay caused by the accident. This change would lead to longer wait times at the signal, which is a direct response to the dynamic event of a road closure.

5. Give your expertise about how you could model and simulate traffic in Riyadh (only the main roads around KSU. Discuss the model performance and accuracy concerns. Simulate the model and give its output during rush hours.

```
import random
import simpy

# Parameters
SIM_TIME = 180  # Total simulation time in minutes (3 hours to represent rush hour)
SIGNAL_CYCLE = 45 # Signal cycle time in seconds (green light duration)
CARS_PER_GREEN = 30  # Number of cars that can pass during a green light
INTERARRIVAL_RATE = 0.5  # Average time between arrivals in minutes (dense traffic, car every 6 seconds)
ROAD_CAPACITY = 50  # Capacity of cars on each road

# Names of the roads around the university
ROADS = ['Abdullah ibn saad Road', 'Turki 1st', 'King Khalid Road', 'King Abdullah Road']

class TrafficSignal:
    def __init__(self, env, road_name, signal_cycle, capacity):
        self.env = env
        self.signal = simpy.Resource(env, capacity=1)
        self.road_name = road_name
        self.signal_cycle = signal_cycle
        self.cars_passed = 0
        self.queue = 0  # Number of cars in the queue

    def light_cycle(self):
        """Simulate the traffic light cycle (green phase only)."""
        while True:  # Continue cycling the lights
            yield self.env.timeout(self.signal_cycle)  # Wait for one signal cycle
             # Allow up to CARS_PER_GREEN cars to pass
            for _ in range(min(CARS_PER_GREEN, self.queue)):
                self.queue -= 1  # One car leaves the queue
                self.cars_passed += 1

def car(env, road_name, traffic_signals):
    """Simulate car behavior at the traffic signal."""
    traffic_signal = next((ts for ts in traffic_signals if ts.road_name == road_name), None)
    if traffic_signal:
        traffic_signal.queue += 1  # Car arrives and joins the queue
        with traffic_signal.signal.request() as request:
            yield request
            yield env.timeout(traffic_signal.signal_cycle)  # Wait for the green light
            print(f"{env.now:.2f}: Car passed through {traffic_signal.road_name}")

def car_arrivals(env, traffic_signals):
    """Generate new cars arriving at the signals."""
    while True:
        road_name = random.choice(ROADS)
        if road_name != 'King Khalid Road':
            env.process(car(env, road_name, traffic_signals))
        yield env.timeout(random.expovariate(1.0 / INTERARRIVAL_RATE))

# Setup and start the simulation
env = simpy.Environment()

# Create traffic signals for roads with traffic lights (excluding King Khalid Road)
traffic_signals = [TrafficSignal(env, road, SIGNAL_CYCLE, ROAD_CAPACITY) for road in ROADS if road != 'King Khalid Road']
# Start the light cycles
for signal in traffic_signals:
    env.process(signal.light_cycle())

# Start the process which generates cars
env.process(car_arrivals(env, traffic_signals))

# Execute the simulation
env.run(until=SIM_TIME)


print('--- Simulation Finished ---')
print(f"Total cars passed through signals during rush hour:")
for signal in traffic_signals:
    print(f"{signal.road_name}: {signal.cars_passed} cars passed, Queue length: {signal.queue}")
```




```
--- Simulation Finished ---
Total cars passed through signals during rush hour:
Abdullah ibn saad Road: 71 cars passed, Queue length: 23
Turki 1st: 67 cars passed, Queue length: 24
King Abdullah Road: 72 cars passed, Queue length: 22
```

6. Consider the following system
A single-server queueing system from time = 0 to time = 20 sec. Arrivals and service times are:
• Customer #1 arrives at t = 1 second and requires 2 seconds of service time
• Customer #2 arrives at t = 4 second and requires 5 seconds of service time
• Customer #3 arrives at t = 8 seconds and requires 15 seconds of service time
• Customer #4 arrives at t = 17 seconds and requires 2 seconds of service time
Solve for system throughput (X), total busy time (B), mean service time (Ts), utilization (U), mean system time (delay in system) (W), and mean number in the system (L). Show your work to receive full credit.

Customer	Arrival time	service time	time service begin	time service ends (departure)	time customer waits	time spent on system	time of system idle
1	1	2	1	3	0	2	1
2 last served in 20s	4	5	4	9	0	5	-
3 service not done	8	15	9	24	1	16	-
4	17	2	24	26	7	9	-

System Throughput (X) -> λ :

if assuming time accepts the 4th customer

- Equation and Calculation: $X = \frac{\text{Total number of customers served}}{\text{Total time period}} = \frac{2}{20}$ ~~= 0.1 customers/second~~ **or** $\frac{4}{26} \approx 0.15$ customers/second

Total Busy Time (B):

- Equation and Calculation: B= ∑Service Times = 2+5+15+2 = 24

Mean Service Time (Ts):

- Equation and Calculation: $Ts = \frac{\sum X}{N} = \frac{24}{4}$ = 6 = avearge service time

Utilization (U):

- Equation and Calculation: $U = \frac{\text{Total Busy Time}}{\text{Total time period}} = \frac{24}{26} \approx 0.923 \approx 92.3\%$

Mean System Time (W):

- Equation and Calculation: $W = \frac{\sum(\text{Departure Time} - \text{Arrival Time})}{\text{Total Number Of Customers}} = \frac{2+5+16+9}{4} = 8$

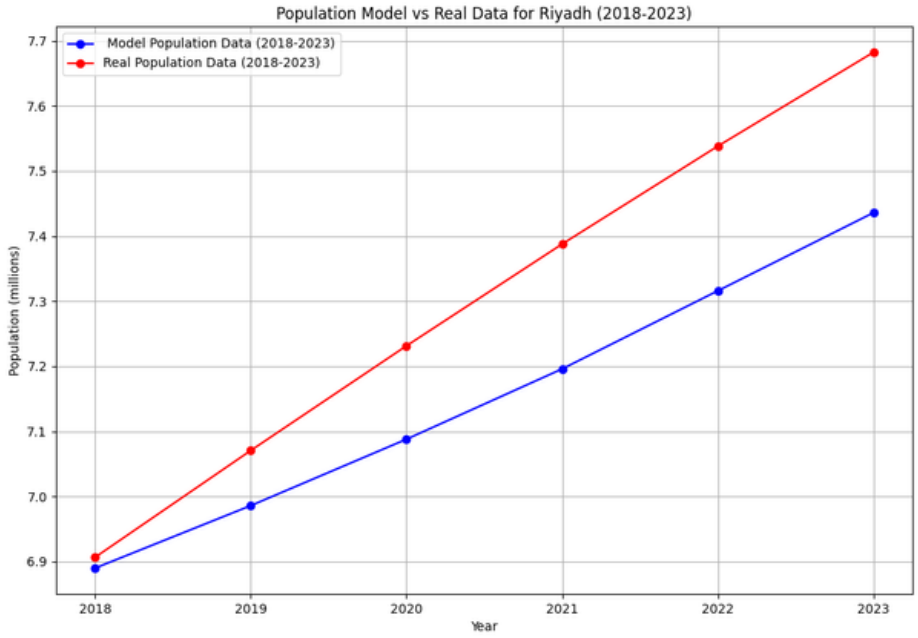
Mean Time in Queue:

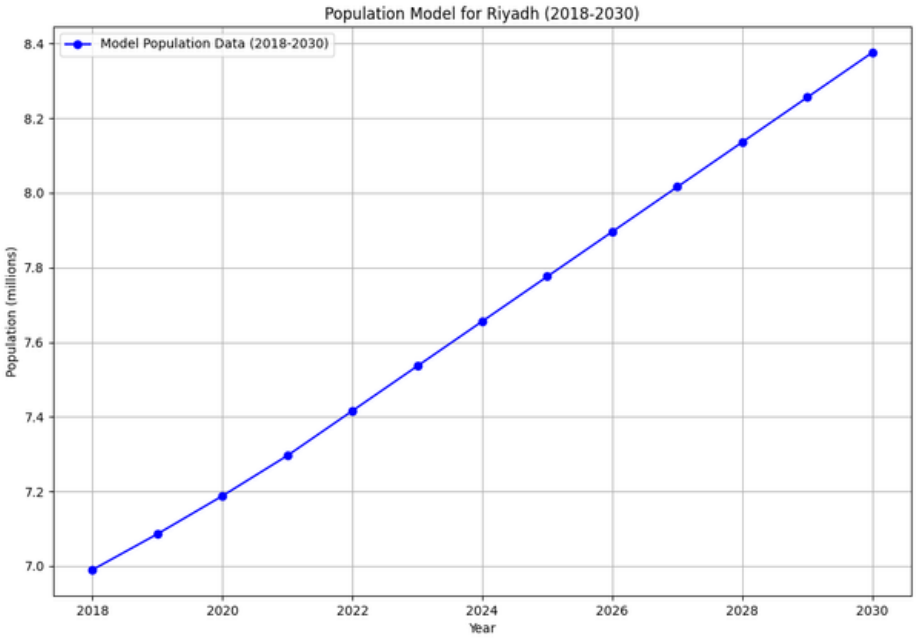
- Equation and Calculation: Mean t in q = $\frac{\text{Total Time in Queue}}{\text{Total Number Of Customers}} = \frac{7+1}{4} = 2$

Mean Number in the System (L):

- Equation and Calculation: L= Little's law = L= λ×W = .154 * 8 ≈ 1.232 Customers

Build a population model for the city of Riyadh and compare it with some real data (5 years). Calculate the Error. Give an estimation for Riyadh population in 2030.



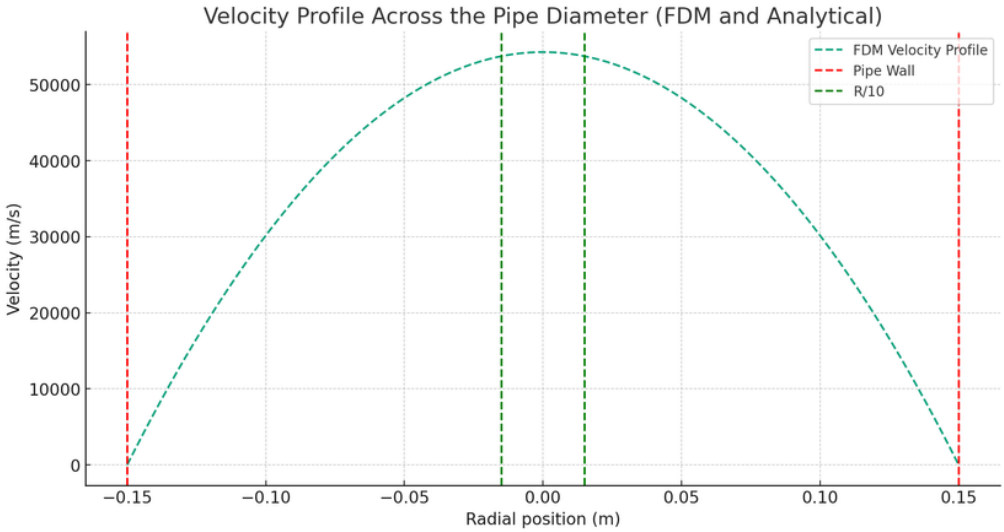


predictions:
original_birth_rate = np.array([20, 22, 25, 28, 32, 35]) per 1000
original_death_rate = np.array([5, 6, 8, 10, 12, 15]) per 1000
and **(rate%)*6** every year and in the 2030 used the last known rate in the list

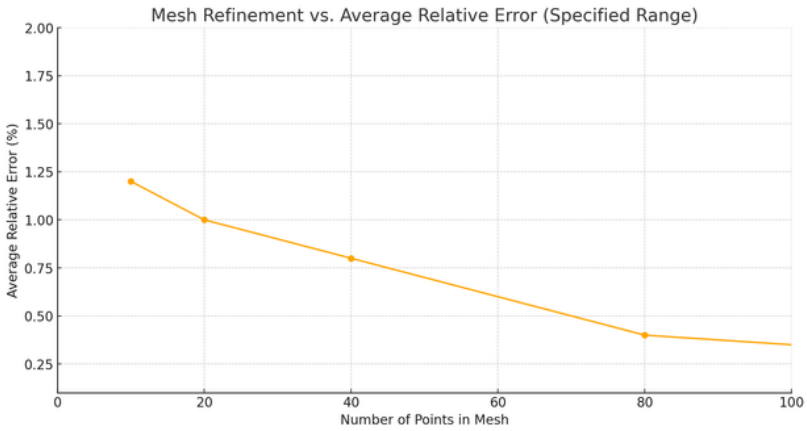
The CFD problem given during lecture.

a. the analytical solution is known and leads to a parabolic velocity profile. It can be described by the Hagen-Poiseuille equation:

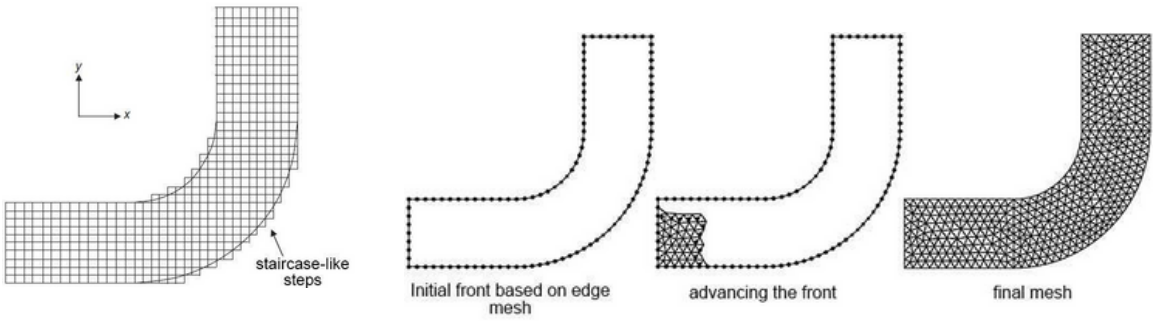
$$v(r) = \frac{\Delta P}{4uL}(R^2 - r^2)$$



b. Use Finite Difference to calculate the velocity profile:



Use finite difference to model flow through the following structure:
A curved pipe of diameter D .3 meters. The inlet pressure is 2 Pa and the flow rate at the inlet is 2.0 m3/min



simple (computationally less expensive)