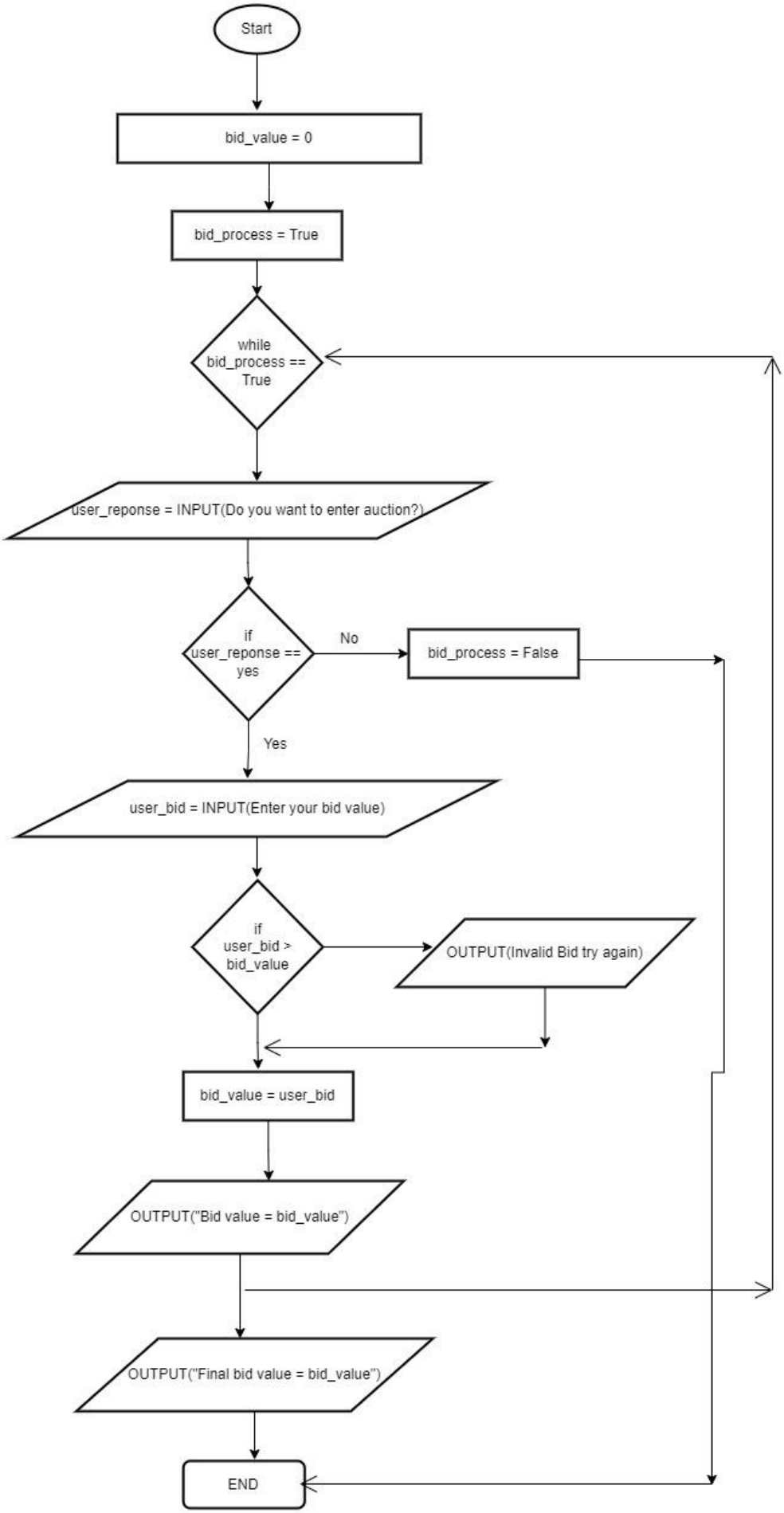
# ARTIFICIAL INTELLIGENCE COURSE WORK

## Year 2 Semester 01

## CM 2602

Dineth Hasaranga – 20210537 - 2117526

(Question 1)(a)(01)



(02) **Problems in Naïve Auction Algorithm:**

1. According to the flow chart drawn if a bidder inputs a higher value than the existing bid value then it will be considered the highest value and ultimately taken as the final value. Though technically its correct if there are more bidders to bid, they will miss the opportunity. So, this can be considered as a problem.

So, what we can do is to create a list of bid values first and iterate over the list to find the final bid value.

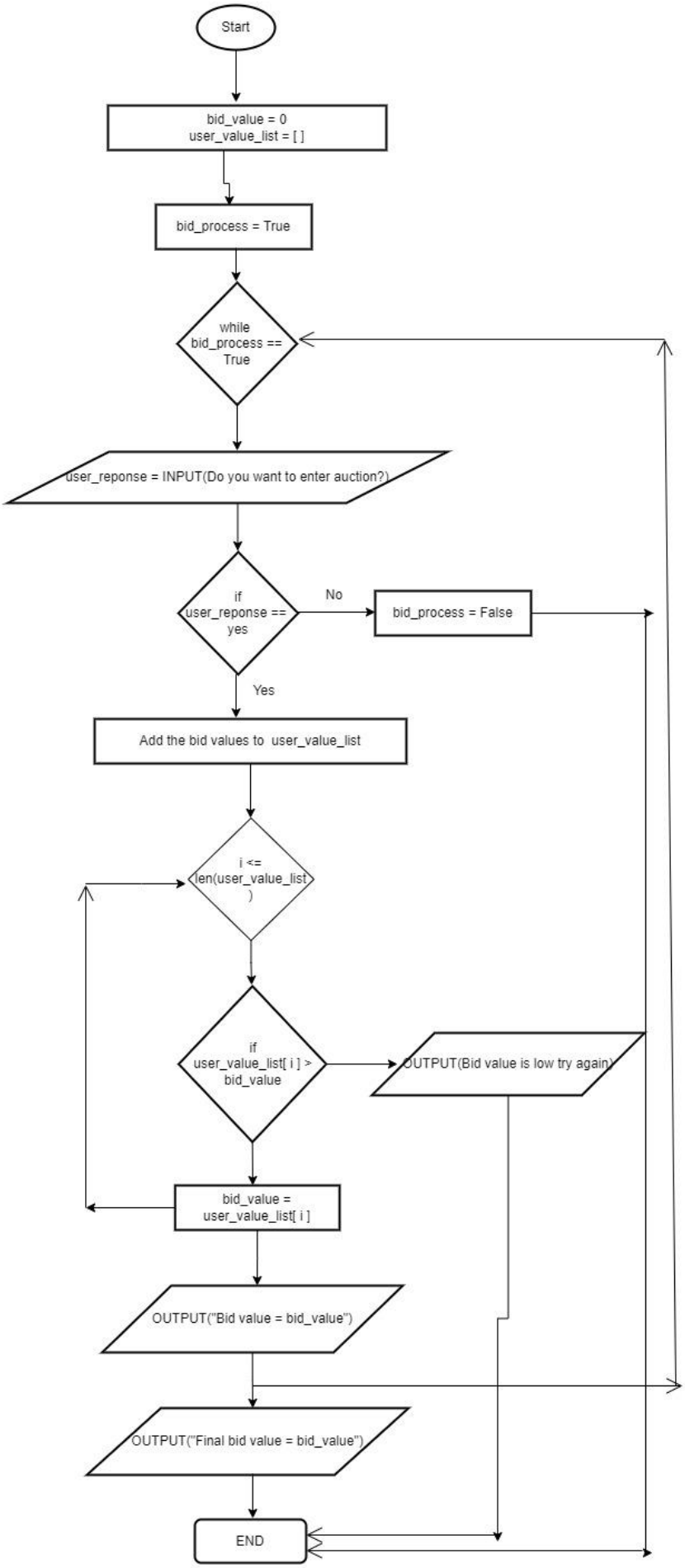
2. Naïve auction algorithm cannot be considered as most efficient algorithm for large scale systems since it is time consuming.

3. Naïve auction algorithm considers only the bid values that are given as inputs and doesn't consider other factors relevant to resource allocation such as the priority of various resources and relative importance of the resources.

So, what we can do is to create a priority order for the resources and check it before the bidding process starts.

- The flow chart with the list of bidders is below. This can overcome the problem 1 mentioned above.

(03)

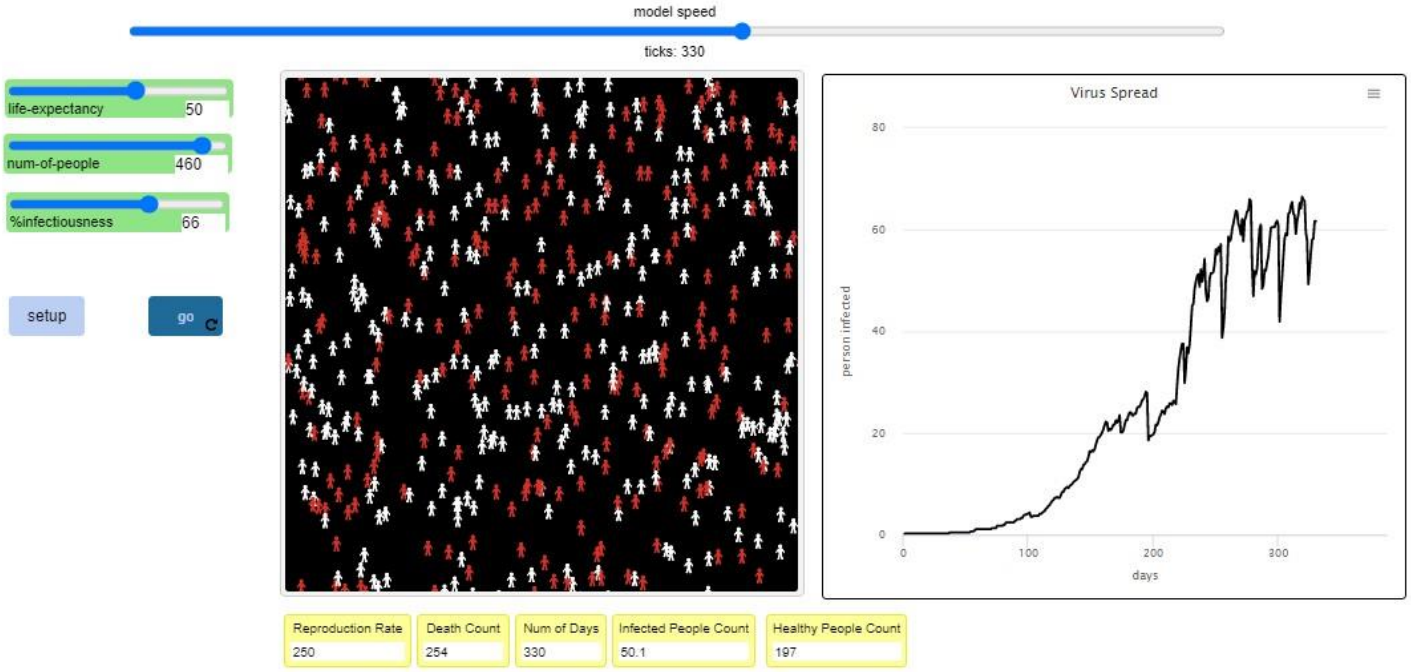

Start
bid_value = 0
user_value_list = [ ]
bid_process = True
while bid_process == True
user_reponse = INPUT(Do you want to enter auction?)
if user_reponse == yes
No
bid_process = False
Yes
Add the bid values to user_value_list
i <= len(user_value_list )
if user_value_list[ i ] > bid_value
OUTPUT(Bid value is low try again)
bid_value = user_value_list[ i ]
OUTPUT("Bid value = bid_value")
OUTPUT("Final bid value = bid_value")
END

(b)(01)

```
globals [%infected
         healthy-people-count
         reproduction-probability
         dead-count]

turtles-own[age-of-person]

; setting the interface to start position
to setup
  clear-all
  reset-ticks
  create-turtles num-of-people
  [ setxy random-xcor random-ycor
    set shape "person"
    set color white
  ]
  ask turtle 1 [set color red]
  set %infected (count turtles with [color = red] / count turtles) * 100
end

; execution code
to go
  tick
  ask turtles [
    rt random 100 lt random 100 fd 1
    age-calculation]
  ask turtles with [color = red]
  [
    ask other turtles-here [if random 100 < %infectiousness [set color red]]
  ]
  set %infected (count turtles with [color = red] / count turtles) * 100
  healthy-people
  recovery-count
  death-calculation
  death-calculation
  reproduction
  if %infected = 100 [stop]
end

; code to healthy people
to healthy-people
  set healthy-people-count (num-of-people - count turtles with [color = red])
end

; code to count death rate
to death-calculation
  set dead-count random 300
  ask turtles with [color = red]
  [
    ask other turtles-here [if dead-count < %infected [die]]
  ]
end

; code to count recovery rate
to recovery-count
  ask turtles with [color = red]
  [
    ask other turtles-here [if random 300 < %infected [set color white]]
  ]
end

; code to reproduce new people
to reproduction
  set reproduction-probability random 300
  if num-of-people > (count turtles)
    [create-turtles reproduction-probability
  [ setxy random-xcor random-ycor
    set shape "person"
```

```
; code to reproduce new people
to reproduction
  set reproduction-probability random 300
  if num-of-people > (count turtles)
    [create-turtles reproduction-probability
  [ setxy random-xcor random-ycor
    set shape "person"
    set color white
  ]]

end

; code to calculate the age
to age-calculation
  set age-of-person age-of-person + 1
  if age-of-person > (life-expectancy * 365) [ die ]
end
```

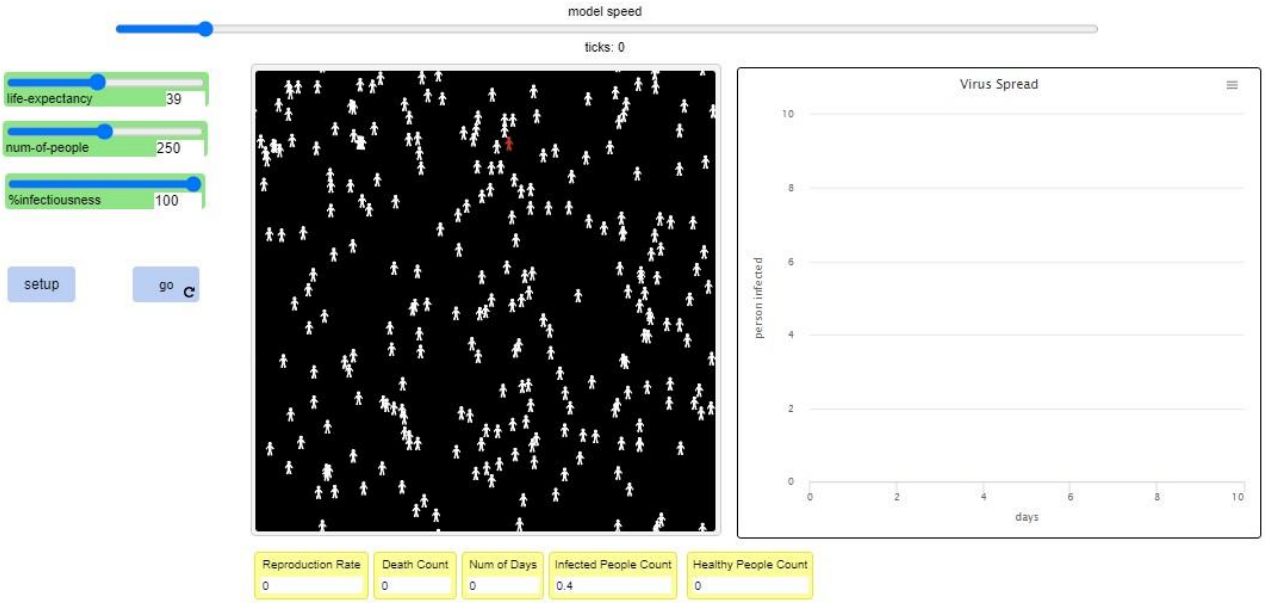
(02)



(03) Screen shot of the code is mentioned above in Q1.

Code snippet 1 – **to setup**

Use to set up the interface with initial requirements for the program to start with.




Code snippet 2 – **to go**

- Use to start the model and activate the other coded functions of the model. Also it defines the movement of the turtle during the process.

Code snippet 3 – **to healthy people**

- Use to calculate the number of healthy people in the model.

Code snippet 4 – **to death-calculation**

- Use to calculate the death rate.

Code snippet 5 – **to recovery-count**

- Use to calculate the number of people recovered.

Code snippet 6 – **to reproduction**

- To produce new people after death and show those people in white color.

Code snippet 6 – **to age calculation**

- To calculate the age of a person.

(Question 2)

(01)

**Scope definition:**

The scope covered by the knowledge graph is the variants of corona virus, infection possibilities of the virus, the symptoms of various kind of the virus and the prevention methods that can be practiced.

**Competency questions:**

1.What are the types of corona virus?

2.What are the types of covid19?

3.What are the types of covid19 virus that can be treated by vaccines?

4.What are the types of covid19 virus that is spread by the wind?

5.What are the precautions that can be taken to protect from the virus?

6.What are the symptoms of the virus?

7.What are the infection possibilities?

**References:**

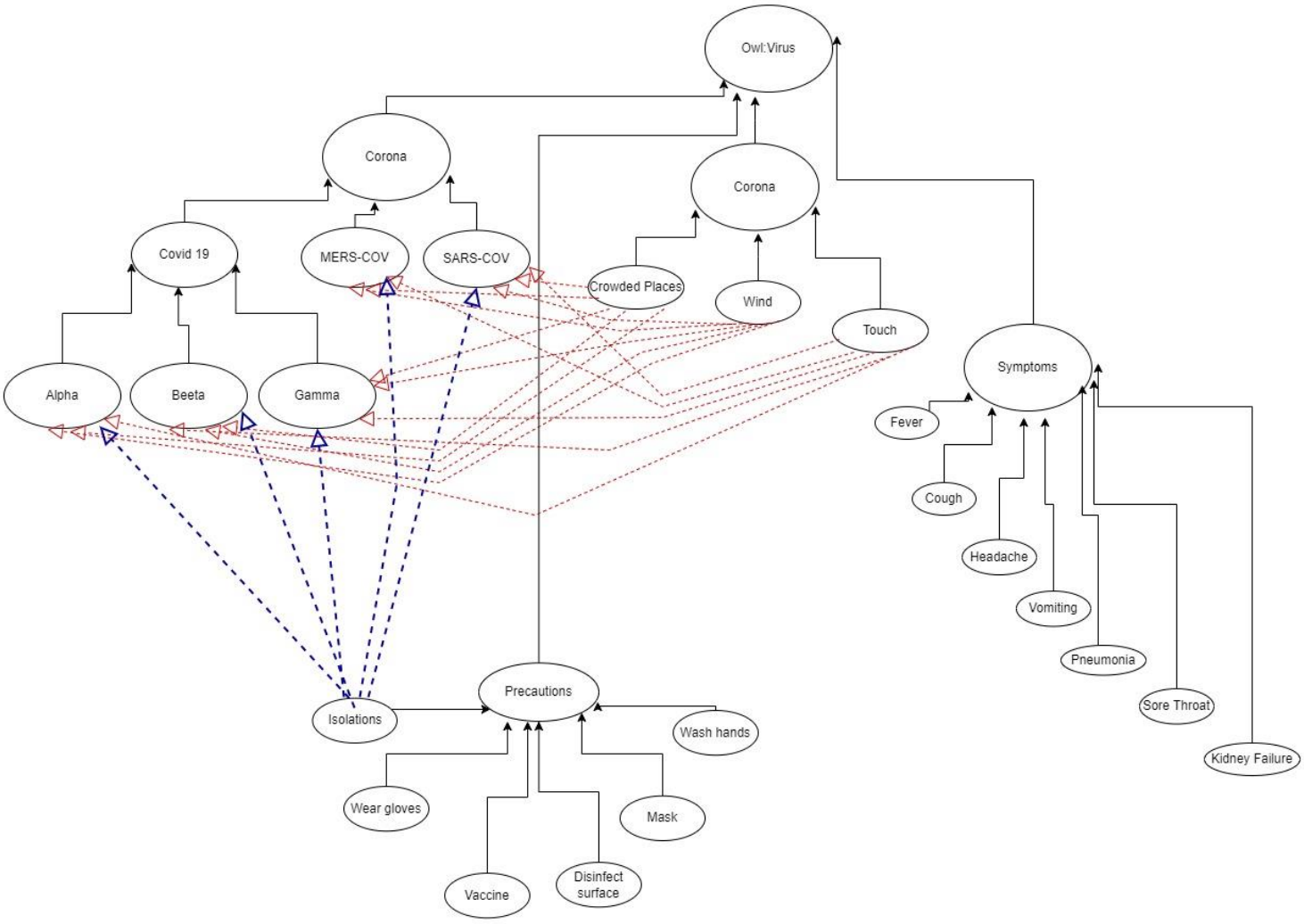

1. CDC and CDC (2022). *Isolation*. [online] Centers for Disease Control and Prevention. Available at: https://www.cdc.gov/coronavirus/2019-ncov/your-health/isolation.html.

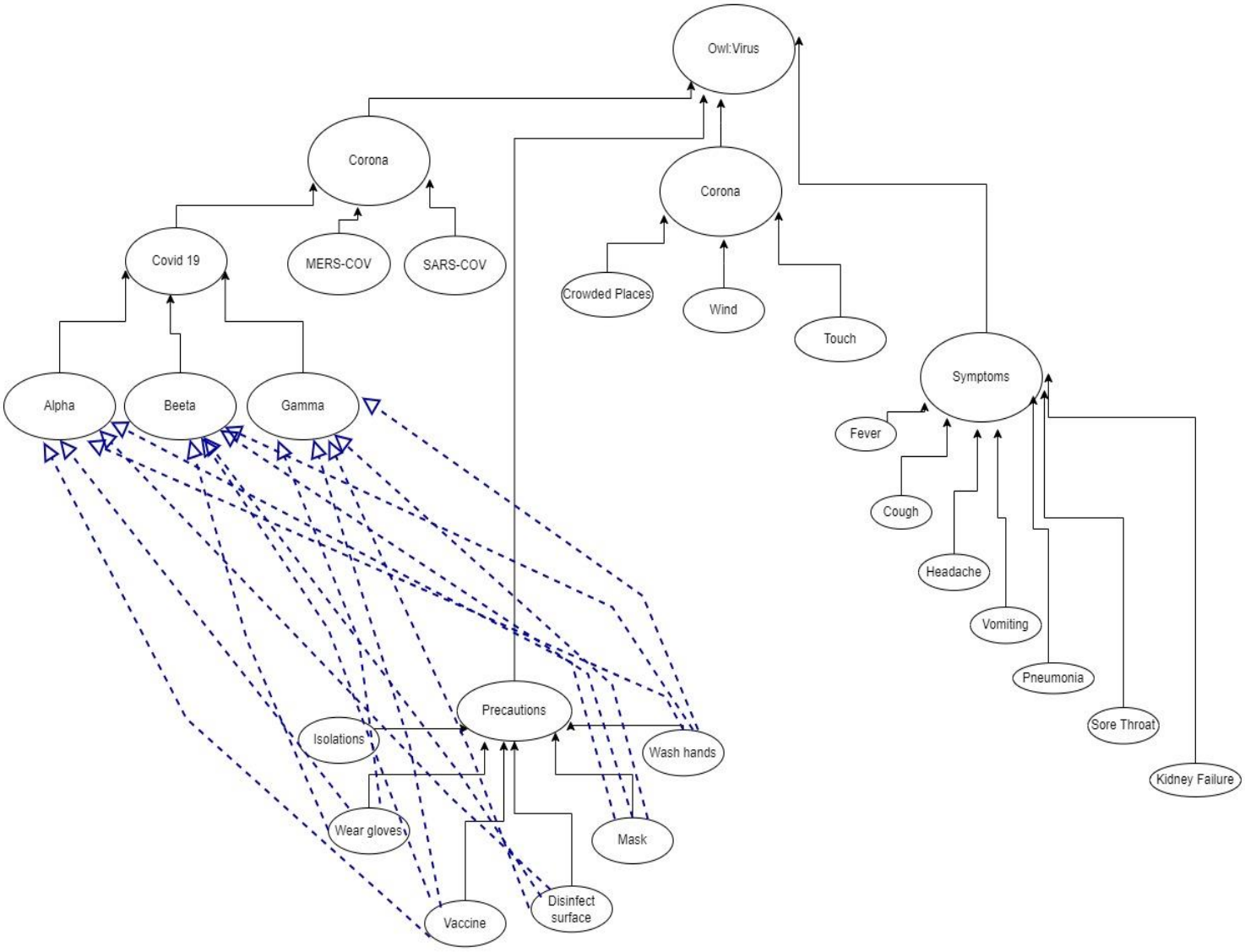
2. Struyf, T., Deeks, J.J., Dinnes, J., Takwoingi, Y., Davenport, C., Leeflang, M.M., Spijker, R., Hooft, L., Emperador, D., Domen, J., Tans, A., Janssens, S., Wickramasinghe, D., Lannoy, V., Horn, S.R.A. and Van den Bruel, A. (2022). Signs and symptoms to determine if a patient presenting in primary care or hospital outpatient settings has COVID-19. *Cochrane Database of Systematic Reviews*, 2022(5). doi:10.1002/14651858.cd013665.pub3.

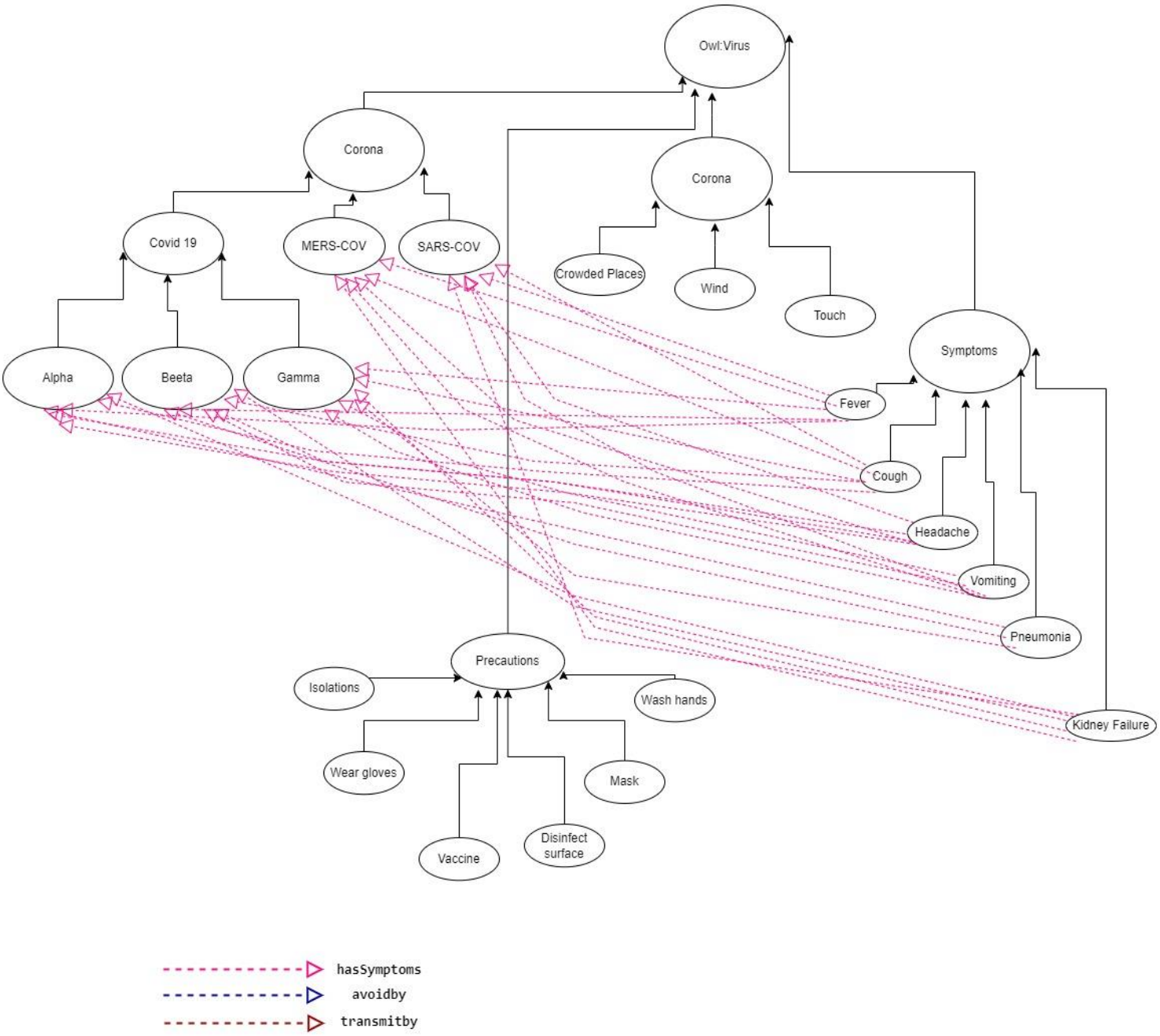
3. Nguyen, T., Duong Bang, D. and Wolff, A. (2020). 2019 Novel Coronavirus Disease (COVID-19): Paving the Road for Rapid Detection and Point-of-Care Diagnostics. *Micromachines*, 11(3), p.306. doi:10.3390/mi11030306.

4. www.who.int. (n.d.). *Coronavirus disease (COVID-19): Ventilation and air conditioning*. [online] Available at: https://www.who.int/news-room/questions-and-answers/item/coronavirus-disease-covid-19-ventilation-and-air-conditioning.

(02) I have used 3 diagrams because it was hard to show the interactions using one diagram. The main graph is same for all 3 diagrams only the interactions are different.




Owl:Virus
Corona
Corona
Covid 19
MERS-COV
SARS-COV
Crowded Places
Wind
Touch
Symptoms
Alpha
Beeta
Gamma
Fever
Cough
Headache
Vomiting
Pneumonia
Sore Throat
Kidney Failure
Precautions
Isolations
Wash hands
Wear gloves
Mask
Disinfect surface
Vaccine


Owl:Virus
Corona
Corona
Covid 19
MERS-COV
SARS-COV
Crowded Places
Wind
Touch
Alpha
Beeta
Gamma
Symptoms
Fever
Cough
Headache
Vomiting
Pneumonia
Kidney Failure
Precautions
Isolations
Wash hands
Wear gloves
Mask
Vaccine
Disinfect surface
hasSymptoms
avoidby
transmitby

(03)

```
<rdf:RDF
xmlns:Virus="http://www.example.com/Virus/Corona.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
>

<!-- OWL Header -->
<owl:Ontology rdf:about="http://www.example.com/Virus">
    <dc:title>Corona Ontology</dc:title>
    <dc:description>A virus ontology </dc:description>
</owl:Ontology>

<!-- Define property -->
<owl:ObjectProperty rdf:about="http://www.example.com/Virus#transmitBy"/>
<owl:ObjectProperty rdf:about="http://www.example.com/Virus#avoidBy"/>
<owl:ObjectProperty rdf:about="http://www.example.com/Virus#hasSymptoms"/>

<!-- OWL Class Definition - Corona Type -->
<owl:Class rdf:about="http://www.example.com/Virus#Corona">
    <rdfs:label>Corona</rdfs:label>
    <rdfs:comment>corona types</rdfs:comment>
</owl:Class>

<!-- OWL Class Definition - Infection Posibility Type -->
<owl:Class rdf:about="http://www.example.com/Virus#infectionPosibilities">
    <rdfs:label>Posibilities</rdfs:label>
    <rdfs:comment>posibilities types</rdfs:comment>
</owl:Class>

<!-- OWL Class Definition - Method of Precautions -->
<owl:Class rdf:about="http://www.example.com/Virus#Precautions">
    <rdfs:label>Precautions</rdfs:label>
    <rdfs:comment>precautions types</rdfs:comment>
</owl:Class>

<!-- OWL Class Definition - Type of Symptoms  -->
<owl:Class rdf:about="http://www.example.com/Virus#Symptoms">
    <rdfs:label>Symptoms</rdfs:label>
    <rdfs:comment>symptoms types</rdfs:comment>
</owl:Class>


<!-- OWL SubClass Definition - Covid19 -->
<owl:Class rdf:about="http://www.example.com/Virus#covid19">

    <!-- Covid19 is a subclassification of Corona -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Corona"/>
```

```
    <rdfs:label>Covid19</rdfs:label>
    <rdfs:comment>covid19 Virus</rdfs:comment>

</owl:Class>

<!-- OWL SubClass Definition - Alpha -->
<owl:Class rdf:about="http://www.example.com/Virus#alpha">

    <!-- Alpha is a subclassification of Corona -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#covid19"/>

    <rdfs:label>Alpha</rdfs:label>
    <rdfs:comment>Alpha description</rdfs:comment>

    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#wind"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#touch"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#crowdedPlaces"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#masks"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Isolation"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Vaccine"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#wearGlows"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#washHands"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#disinfectSurface"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#fever"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#cough"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#headache"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#vomiting"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#kidneyFailure"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#pneumonia"/>
        </owl:Restriction>
    </rdfs:subClassOf>

</owl:Class>

<!-- OWL SubClass Definition - Beta -->
<owl:Class rdf:about="http://www.example.com/Virus#beta">

    <!-- Beta is a subclassification of Corona -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#covid19"/>

    <rdfs:label>Beta</rdfs:label>
    <rdfs:comment>Beta description</rdfs:comment>

    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#wind"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#touch"/>
```

```
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#crowdedPlaces"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#masks"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Isolation"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Vaccine"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#wearGlows"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#washHands"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#disinfectSurface"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#fever"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#cough"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#headache"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#vomiting"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#kidneyFailure"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#pneumonia"/>
        </owl:Restriction>
    </rdfs:subClassOf>

</owl:Class>

<!-- OWL SubClass Definition - Gamma -->
<owl:Class rdf:about="http://www.example.com/Virus#gamma">

    <!-- Gamma is a subclassification of Corona -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#covid19"/>

    <rdfs:label>Gamma</rdfs:label>
    <rdfs:comment>Gamma description</rdfs:comment>

    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#wind"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#touch"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#crowdedPlaces"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#masks"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Isolation"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Vaccine"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#wearGlows"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#washHands"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#disinfectSurface"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#fever"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#cough"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#headache"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#vomiting"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#kidneyFailure"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#pneumonia"/>
        </owl:Restriction>
    </rdfs:subClassOf>

</owl:Class>
```

```
<!-- OWL SubClass Definition - MERS -->
<owl:Class rdf:about="http://www.example.com/Virus#mersCov">

    <!-- MERS is a subclassification of Corona -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Corona"/>

    <rdfs:label>MERS</rdfs:label>
    <rdfs:comment>MERS description</rdfs:comment>

    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#wind"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#touch"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#crowdedPlaces"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#masks"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Isolation"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Vaccine"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#wearGlows"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#washHands"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#fever"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#cough"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#headache"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#vomiting"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#kidneyFailure"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#septicShock"/>
        </owl:Restriction>
    </rdfs:subClassOf>

</owl:Class>


<!-- OWL SubClass Definition - SARS -->
<owl:Class rdf:about="http://www.example.com/Virus#sarsCov">

    <!-- SARS is a subclassification of Corona -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Corona"/>

    <rdfs:label>SARS</rdfs:label>
    <rdfs:comment>SARS description</rdfs:comment>

    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#wind"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#touch"/>
            <owl:transmitBy rdf:resource="http://www.example.com/Virus#crowdedPlaces"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#masks"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Isolation"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#Vaccine"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#wearGlows"/>
            <owl:avoidBy rdf:resource="http://www.example.com/Virus#washHands"/>
```

```
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#fever"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#cough"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#headache"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#vomiting"/>
            <owl:hasSymptoms rdf:resource="http://www.example.com/Virus#kidneyFailure"/>
        </owl:Restriction>
    </rdfs:subClassOf>

</owl:Class>

<!-- OWL SubClass Definition - Wind -->
<owl:Class rdf:about="http://www.example.com/Virus#wind">

    <!-- Masks is a subclassification of Infection Posibility -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#infectionPosibilities"/>

    <rdfs:label>Wind</rdfs:label>
    <rdfs:comment>Wind</rdfs:comment>

</owl:Class>

<!-- Define the Wind class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#wind">

    <!-- Wind is an individual (instance) of the Infection Posibility class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#infectionPosibilities"/>

</rdf:description>

<!-- OWL SubClass Definition - Touch -->
<owl:Class rdf:about="http://www.example.com/Virus#touch">

    <!-- Masks is a subclassification of Infection Posibility -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#infectionPosibilities"/>

    <rdfs:label>Touch</rdfs:label>
    <rdfs:comment>Touch</rdfs:comment>

</owl:Class>

<!-- Define the Touch class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#touch">

    <!-- Touch is an individual (instance) of the Infection Posibility class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#infectionPosibilities"/>

</rdf:description>

<!-- OWL SubClass Definition - Crowded Places -->
<owl:Class rdf:about="http://www.example.com/Virus#crowdedPlaces">
```

```
    <!-- Masks is a subclassification of Infection Posibility -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#infectionPosibilities"/>

    <rdfs:label>Crowded Places</rdfs:label>
    <rdfs:comment>Crowded Places</rdfs:comment>

</owl:Class>

<!-- Define the Crowded Places class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#crowdedPlaces">

    <!-- Crowded Places is an individual (instance) of the Infection Posibility class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#infectionPosibilities"/>

</rdf:description>

<!-- OWL SubClass Definition - Masks -->
<owl:Class rdf:about="http://www.example.com/Virus#masks">

    <!-- Masks is a subclassification of Precautions -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Precautions"/>

    <rdfs:label>Masks</rdfs:label>
    <rdfs:comment>Masks</rdfs:comment>

</owl:Class>

<!-- Define the Masks class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#masks">

    <!-- Masks is an individual instance of the Precaution class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Precautions"/>

</rdf:description>

<!-- OWL SubClass Definition - Isolation -->
<owl:Class rdf:about="http://www.example.com/Virus#Isolation">

    <!-- Isolation is a sub-classification of Precaution Class -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Precautions"/>

    <rdfs:label>Isolation</rdfs:label>
    <rdfs:comment>Isolation</rdfs:comment>

</owl:Class>

<!-- Define the Isolation class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#Isolation">
```

```
    <!-- Isolation is an individual instance of the Precaution class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Precautions"/>

</rdf:description>

<!-- OWL SubClass Definition - Vaccine -->
<owl:Class rdf:about="http://www.example.com/Virus#Vaccine">

    <!-- Vaccine is a sub-classification of the Precaution Class -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Precautions"/>

    <rdfs:label>Vaccine</rdfs:label>
    <rdfs:comment>Vaccine</rdfs:comment>

</owl:Class>

<!-- Define the Vaccine class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#Vaccine">

    <!-- Vaccine is an individual instance of Precaution class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Precautions"/>

</rdf:description>

<!-- OWL SubClass Definition - Wear Glows -->
<owl:Class rdf:about="http://www.example.com/Virus#wearGlows">

    <!-- Wear Glows is a sub-classification of the Precaution Class -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Precautions"/>

    <rdfs:label>Wear Glows</rdfs:label>
    <rdfs:comment>Wear Glows</rdfs:comment>

</owl:Class>

<!-- Define the Wear Glows class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#wearGlows">

    <!-- Wear Glows is an individual instance of the Precaution class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Precautions"/>

</rdf:description>

<!-- OWL SubClass Definition - Wash Hands -->
<owl:Class rdf:about="http://www.example.com/Virus#washHands">

    <!-- Wash Hands is a sub-classification of the Precaution class -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Precautions"/>

    <rdfs:label>Wash Hands</rdfs:label>
```

```
    <rdfs:comment>Wash Hands</rdfs:comment>

</owl:Class>

<!-- Define the Wash Hands class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#washHands">

    <!-- Wash Hands is an individual instance of Precaution class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Precautions"/>

</rdf:description>

<!-- OWL SubClass Definition - Disinfect Surface -->
<owl:Class rdf:about="http://www.example.com/Virus#disinfectSurface">

    <!-- Disinfect Surface is a sub-classification of the Precaution class -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Precautions"/>

    <rdfs:label>Disinfect Surface</rdfs:label>
    <rdfs:comment>Disinfect Surface</rdfs:comment>

</owl:Class>

<!-- Define the Disinfect Surface class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#disinfectSurface">

    <!-- Disinfect Surface is an individual instance of Precaution class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Precautions"/>

</rdf:description>

<!-- OWL SubClass Definition - Fever -->
<owl:Class rdf:about="http://www.example.com/Virus#fever">

    <!-- Fever is a sub-classification of Symptoms -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>

    <rdfs:label>Fever</rdfs:label>
    <rdfs:comment>Fever</rdfs:comment>

</owl:Class>

<!-- Define the Fever class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#fever">

    <!-- Fever is an individual instance of Symptoms class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>
```

```
<!-- OWL SubClass Definition - Cough -->
<owl:Class rdf:about="http://www.example.com/Virus#cough">

    <!-- Cough is a subclassification of Symptoms -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>

    <rdfs:label>Cough</rdfs:label>
    <rdfs:comment>Cough</rdfs:comment>

</owl:Class>

<!-- Define the Cough class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#cough">

    <!-- Cough is an individual instance of Symptoms class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>

<!-- OWL SubClass Definition - Headache -->
<owl:Class rdf:about="http://www.example.com/Virus#headache">

    <!-- Headache is a sub-classification of Symptom -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>

    <rdfs:label>Headache</rdfs:label>
    <rdfs:comment>Headache</rdfs:comment>

</owl:Class>

<!-- Define the Headache class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#headache">

    <!-- Headache is an individual instance of Symptom class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>

<!-- OWL SubClass Definition - Vomiting -->
<owl:Class rdf:about="http://www.example.com/Virus#vomiting">

    <!-- Vomiting is a sub-classification of Symptom class -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>

    <rdfs:label>Vomiting</rdfs:label>
    <rdfs:comment>Vomiting</rdfs:comment>

</owl:Class>

<!-- Define the Vomiting class instance -->
```

```
<rdf:description rdf:about="http://www.example.com/Virus#vomiting">

    <!-- Vomiting is an individual instance of Symptom class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>

<!-- OWL SubClass Definition - Kidney Failure -->
<owl:Class rdf:about="http://www.example.com/Virus#kidneyFailure">

    <!-- Kidney Failure is a sub-classification of the Symptom -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>

    <rdfs:label>Kidney Failure</rdfs:label>
    <rdfs:comment>Kidney Failure</rdfs:comment>

</owl:Class>

<!-- Define the Kidney Failure class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#kidneyFailure">

    <!-- Kidney Failure is an individual instance of Symptom class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>

<!-- OWL SubClass Definition - Septic Shock -->
<owl:Class rdf:about="http://www.example.com/Virus#septicShock">

    <!-- Septic Shock is a sub-classification of the Symptom -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>

    <rdfs:label>Septic Shock</rdfs:label>
    <rdfs:comment>Septic Shock</rdfs:comment>

</owl:Class>

<!-- Define the Septic Shock class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#septicShock">

    <!-- Septic Shock is an individual instance of Symptom class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>

<!-- OWL SubClass Definition - Pneumonia -->
<owl:Class rdf:about="http://www.example.com/Virus#pneumonia">

    <!-- Pneumonia is a sub-classification of the Symptom -->
    <rdfs:subClassOf rdf:resource="http://www.example.com/Virus#Symptoms"/>
```

```
    <rdfs:label>Pneumonia</rdfs:label>
    <rdfs:comment>Pneumonia</rdfs:comment>

</owl:Class>

<!-- Define the Pneumonia class instance -->
<rdf:description rdf:about="http://www.example.com/Virus#pneumonia">

    <!-- pneumonia is an individual instance of Symptom class -->
    <rdf:type rdf:resource="http://www.example.com/Virus#Symptoms"/>

</rdf:description>

</rdf:RDF>
```

(04)

1.**What are the types of corona virus?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:Corona
}
```

```
1  <http://www.example.com/Virus#covid19>
2  <http://www.example.com/Virus#mersCov>
3  <http://www.example.com/Virus#sarsCov>
```

2. **What are the types of covid19?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:covid19
}
```

```
1  <http://www.example.com/Virus#alpha>
2  <http://www.example.com/Virus#beta>
3  <http://www.example.com/Virus#gamma>
```

3.**What are the types of covid19 virus that can be treated by vaccines?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:covid19 .
  ?x rdfs:subClassOf [
        a owl:Restriction ;
        owl:avoidBy Virus:Vaccine;
    ]
}
```

```
1  <http://www.example.com/Virus#alpha>
2  <http://www.example.com/Virus#beta>
3  <http://www.example.com/Virus#gamma>
```

4. **What are the types of covid19 virus that is spread by the wind?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:covid19 .
  ?x rdfs:subClassOf [
        a owl:Restriction ;
        owl:transmitBy Virus:wind;
    ]
}
```

```
1  <http://www.example.com/Virus#alpha>
2  <http://www.example.com/Virus#beta>
3  <http://www.example.com/Virus#gamma>
```

5.**What are the precautions that can be taken to protect from the virus?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:Precautions
}
```

| | |
|---|---|
| 1 | <http://www.example.com/Virus#masks> |
| 2 | <http://www.example.com/Virus#Isolation> |
| 3 | <http://www.example.com/Virus#Vaccine> |
| 4 | <http://www.example.com/Virus#wearGlows> |
| 5 | <http://www.example.com/Virus#washHands> |
| 6 | <http://www.example.com/Virus#disinfectSurface> |

6.**What are the symptoms of the virus?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:Symptoms
}
```

| | |
|---|---|
| 1 | <http://www.example.com/Virus#fever> |
| 2 | <http://www.example.com/Virus#cough> |
| 3 | <http://www.example.com/Virus#headache> |
| 4 | <http://www.example.com/Virus#vomiting> |
| 5 | <http://www.example.com/Virus#kidneyFailure> |
| 6 | <http://www.example.com/Virus#pneumonia> |
| 7 | <http://www.example.com/Virus#septicShock> |

7. **What are the infection possibilities?**

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX Virus: <http://www.example.com/Virus#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?x
WHERE {
  ?x rdfs:subClassOf Virus:infectionPosibilities
}
```

| | |
|---|---|
| 1 | <http://www.example.com/Virus#wind> |
| 2 | <http://www.example.com/Virus#touch> |
| 3 | <http://www.example.com/Virus#crowdedPlaces> |

(QUESTION 03) **TASK 01/ TASK 02/ TASK 03/ TASK 04**

```
import random as rand

# Creating the maze - Task 01
random_maze = []

maze_rows = 6
maze_columns = 6

for i in range(maze_rows):
    line = []
    for j in range(maze_columns):
        line.append(0)
    random_maze.append(line)

# Creating the start cell of the maze
start_cell_row = rand.randint(0, 1)
start_cell_column = rand.randint(0, 5)
random_maze[start_cell_column][start_cell_row] = 7

# Creating the goal cell of the maze
goal_cell_row = rand.randint(4, 5)
goal_cell_column = rand.randint(0, 5)
random_maze[goal_cell_column][goal_cell_row] = 8

# Creating the barrier cells of the maze
barrier_counter = 0
while barrier_counter < 4:
    barrier_row = rand.randint(0, 5)
    barrier_column = rand.randint(0, 5)

    if random_maze[barrier_column][barrier_row] != 0:
        continue
    random_maze[barrier_column][barrier_row] = 1
    barrier_counter = barrier_counter + 1

for i in random_maze:
    print(i)
print("\n")

# DSF SEARCH ALOGRITHM - Task 02
# Defining the directions allowed to move - up/down/left/right/diagonal

def valid_moves(x_cordinate, y_cordinate, maze):

    next_valid_move = []

#Search directions

    west_search = maze[y_cordinate][x_cordinate - 1]

    if y_cordinate < 5:
        north_search = maze[y_cordinate + 1][x_cordinate]

    if x_cordinate < 5:
        east_search = maze[y_cordinate][x_cordinate + 1]

    south_search = maze[y_cordinate - 1][x_cordinate]

    if y_cordinate < 5:
        northwest_search = maze[y_cordinate + 1][x_cordinate - 1]
```

```
    if y_cordinate < 5 and x_cordinate < 5:
        northeast_search = maze[y_cordinate + 1][x_cordinate + 1]

    if x_cordinate < 5:
        southeast_search = maze[y_cordinate - 1][x_cordinate + 1]

    southwest_search = maze[y_cordinate - 1][x_cordinate - 1]

#Search process

    if x_cordinate-1>=0 and west_search!=1:
        next_valid_move.append((y_cordinate,x_cordinate - 1))

    if y_cordinate+1<len(maze) and north_search!=1:
        next_valid_move.append((y_cordinate + 1,x_cordinate))

    if x_cordinate+1<len(maze) and east_search!=1:
        next_valid_move.append((y_cordinate,x_cordinate + 1))

    if y_cordinate-1>=0 and south_search!=1:
        next_valid_move.append((y_cordinate - 1,x_cordinate))

    if (x_cordinate - 1 >= 0 and y_cordinate + 1 < len(maze)) and northwest_search!=1:
        next_valid_move.append((y_cordinate + 1,x_cordinate - 1))

    if (x_cordinate + 1 < len(maze) and y_cordinate + 1 < len(maze)) and
northeast_search!=1:
        next_valid_move.append((y_cordinate + 1,x_cordinate + 1))

    if (x_cordinate + 1 < len(maze) and y_cordinate - 1 >= 0) and southeast_search!=1:
        next_valid_move.append((y_cordinate - 1,x_cordinate + 1))

    if (x_cordinate - 1 >= 0 and y_cordinate - 1 >= 0) and southwest_search!=1:
        next_valid_move.append((y_cordinate - 1,x_cordinate - 1))


    return next_valid_move

# Apply DFS search to the maze to find the goal

def DFS_Process(maze, stack, visited_cells):

    start = (start_cell_column,start_cell_row)
    goal = (goal_cell_column,goal_cell_row)
    stack.append(start)

    while stack:
        currentExploreCell = stack.pop()
        # print("Popping", currentExploreCell)

        if currentExploreCell == goal:
            print("Goal found")
            print("This is the Final path", visited_cells)
            Total_Node = len(visited_cells)
            print("Time taken to find the goal: ", Total_Node, "minutes")
            return

        a = currentExploreCell[0]
        b = currentExploreCell[1]
        next_steps = valid_moves(a, b, maze)
        # print("Next steps is", next_steps)

        for next_cell in next_steps:
```

```
            if next_cell in visited_cells:
                # print(next_cell, ":visited cell")
                continue
            visited_cells.add(next_cell)
            stack.append(next_cell)

            # print("Visited nodes:", visited_cells)
            # print("Explore cells", stack)

# "explore_cells_stack" list work as a stack
explore_cells_stack = []
# visited set doesn't contain repeated cells.
visited = set()

valid_moves(start_cell_row, start_cell_column, random_maze)
DFS_Process(random_maze, explore_cells_stack, visited)

print("\n")
print("End of DFS Search Algorithm")
print("\n")
print("Start of A Star Search Algorithm")
print("\n")


# Heuristic cost calculation - Task 03
def Chebyshev_distance_calculation(goal_row_che, goal_col_che):
    chebyshev_value_list = []

    for row_node in range (maze_rows):
        for column_node in range (maze_columns):
            value = max(abs(row_node - goal_row_che), abs(column_node - goal_col_che))
            chebyshev_value_list.append(value)
            column_node += 1
        row_node += 1

    return chebyshev_value_list


# A star search algorithm - Task 04
def a_Serch(insert_Maze, start_row, start_column, goal_row, goal_column):
    cell_value_list = []

    # The below list works as a priority queue
    p_queue = []

    visited_nodes_list = []
    heuristic_value_dictionary = {} # Conatains a dictionary with heuristic values and
respective cell

    startNode = (start_column, start_row)
    goalNode = (goal_column, goal_row)
    childNode = startNode

    # Dictionary to find the path from start to the goal node
    path_reverse_dic = {}

    # Updating heuristic cost to dictionary from heuristic_cost_calculation function
    heuristicValue = Chebyshev_distance_calculation(goal_row, goal_column)
    print("Heuristic Values")
    print(heuristicValue)
    print("\n")

    # Creating the heuristic value dictionary
```

```
    for i in range(6):
        for j in range(6):
            temp_cell = (j, i)
            cell_value_list.append(temp_cell)
    for i in range(36):
        value = cell_value_list[i]
        heuristic_value_dictionary[value] = heuristicValue[i]
    print("Heuristic value and the respective cell")
    print(heuristic_value_dictionary)
    print("\n")

    # Dictionary to hold g(n) values for each cell
    g_score_dict = {cell: float('inf') for cell in cell_value_list}
    g_score_dict[startNode] = 0

    # Dictionary to hold final value(g(n)+h(n)) for each cell
    f_score_dict = {cell: float('inf') for cell in cell_value_list}
    f_score_dict[startNode] = heuristic_value_dictionary[startNode]

    p_queue.append(startNode)

    while len(p_queue) != 0:
        curr_cell_node = p_queue.pop(0)
        start_column = curr_cell_node[0]
        start_row = curr_cell_node[1]
        visited_nodes_list.append(curr_cell_node)

        # Defining the directions allowed to move - up/down/left/right/diagonal
        west_search = insert_Maze[start_column][start_row - 1]

        if start_column < 5:
            north_search = insert_Maze[start_column + 1][start_row]

        if start_row < 5:
            east_search = insert_Maze[start_column][start_row + 1]

        south_search = insert_Maze[start_column - 1][start_row]

        if start_column < 5:
            L = insert_Maze[start_row - 1][start_column + 1] #northwest_search

        if start_column < 5 and start_row < 5:
            M = insert_Maze[start_row + 1][start_column + 1] #northeast_search

        if start_row < 5:
            P = insert_Maze[start_row + 1][start_column - 1] #southeast_search

        O = insert_Maze[start_row - 1][start_column - 1] #southwest_search

        if curr_cell_node == goalNode:
            print("Goal has found")
            # print("Visited nodes")
            print(visited_nodes_list)
            length_node_list = len(visited_nodes_list)
            print("Time taken to find the goal: ", length_node_list, "minutes")
            break
        else:
            for i in "ESNWLMPO": #Search process

                if i == "W":
                    if start_row - 1 >= 0 and west_search!=1:
                        childNode = (start_column, start_row - 1)
```

```
                elif i == "N":
                    if start_column + 1 <len(insert_Maze) and north_search!=1:
                        childNode = (start_column + 1, start_row)

                elif i == "S":
                    if start_column - 1 >0 and south_search!=1:
                        childNode = (start_column - 1, start_row)

                elif i == "E":
                    if start_row + 1 < len(insert_Maze) and east_search!=1:
                        childNode = (start_column, start_row + 1)

                elif i == "L":
                    if (start_row - 1 >= 0 and start_column + 1 < len(insert_Maze)) and
L!=1:
                        childNode = (start_column + 1, start_row - 1)

                elif i == "M":
                    if (start_row + 1 < len(insert_Maze) and start_column + 1 <
len(insert_Maze)) and M!=1:
                        childNode = (start_column + 1, start_row + 1)

                elif i == "P":
                    if (start_row + 1 < len(insert_Maze) and start_column - 1 >= 0) and
P!=1:
                        childNode = (start_column - 1, start_row + 1)

                elif i == "O":
                    if (start_row - 1 >= 0 and start_column - 1 >= 0) and O!=1:
                        childNode = (start_column - 1, start_row - 1)

                temp_g_value = g_score_dict[curr_cell_node] + 1
                temp_final_value = temp_g_value + heuristic_value_dictionary[childNode]

                if temp_final_value < f_score_dict[childNode]:
                    g_score_dict[childNode] = temp_g_value
                    f_score_dict[childNode] = temp_final_value
                    p_queue.append(childNode)
                    path_reverse_dic[childNode] = curr_cell_node

    cell_temp = (goal_column, goal_row)
    path_forward = {}
    while cell_temp != startNode:
        path_forward[path_reverse_dic[cell_temp]] = cell_temp
        cell_temp = path_reverse_dic[cell_temp]

    print("\n")
    print("The final path :")
    print(path_forward.values())

Chebyshev_distance_calculation(goal_cell_row, goal_cell_column)
a_Serch(random_maze, start_cell_row, start_cell_column, goal_cell_row, goal_cell_column)
```

**OUTPUT:**

[0, 0, 0, 0, 8, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[1, 7, 0, 0, 0, 0]
[0, 1, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0]

Goal found

This is the Final path {(4, 0), (3, 4), (4, 3), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (1, 0), (2, 5), (1, 3), (4, 5), (3, 3), (5, 0), (5, 3), (0, 1), (2, 4), (1, 2), (0, 4), (2, 1), (1, 5), (3, 2), (3, 5), (4, 4), (5, 5), (0, 0), (1, 1), (0, 3), (2, 0), (1, 4), (2, 3)}

Time taken to find the goal: 32 minutes

End of DFS Search Algorithm

Start of A Star Search Algorithm

Heuristic Values

[4, 4, 4, 4, 4, 5, 3, 3, 3, 3, 4, 5, 2, 2, 2, 3, 4, 5, 1, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 1, 1, 2, 3, 4, 5]

Heuristic value and the respective cell

{(0, 0): 4, (1, 0): 4, (2, 0): 4, (3, 0): 4, (4, 0): 4, (5, 0): 5, (0, 1): 3, (1, 1): 3, (2, 1): 3, (3, 1): 3, (4, 1): 4, (5, 1): 5, (0, 2): 2, (1, 2): 2, (2, 2): 2, (3, 2): 3, (4, 2): 4, (5, 2): 5, (0, 3): 1, (1, 3): 1, (2, 3): 2, (3, 3): 3, (4, 3): 4, (5, 3): 5, (0, 4): 0, (1, 4): 1, (2, 4): 2, (3, 4): 3, (4, 4): 4, (5, 4): 5, (0, 5): 1, (1, 5): 1, (2, 5): 2, (3, 5): 3, (4, 5): 4, (5, 5): 5}

Goal has found

[(3, 1), (3, 2), (2, 1), (4, 0), (4, 2), (2, 2), (2, 0), (3, 3), (4, 1), (4, 3), (2, 3), (1, 1), (3, 0), (1, 2), (1, 0), (5, 0), (5, 1), (5, 3), (1, 3), (3, 4), (4, 4), (5, 2), (5, 4), (2, 4), (0, 2), (0, 0), (0, 1), (1, 4), (0, 4)]

Time taken to find the goal: 29 minutes

The final path :

dict_values([(0, 4), (1, 3), (2, 2)])

```
[0, 0, 0, 0, 8, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[1, 7, 0, 0, 0, 0]
[0, 1, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0]


 Goal found
 This is the Final path {(4, 0), (3, 4), (4, 3), (3, 1), (5, 4), (5, 1), (0, 2), (0, 5), (2, 2), (1, 0), (2, 5), (1, 3), (4, 5), (3, 3), (5, 0), (5, 3), (0,
 Time taken to find the goal:  32 minutes

End of DFS Search Algorithm


Start of A Star Search Algorithm


Heuristic Values
[4, 4, 4, 4, 4, 5, 3, 3, 3, 3, 4, 5, 2, 2, 2, 3, 4, 5, 1, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 1, 1, 2, 3, 4, 5]


Heuristic value and the respective cell
{(0, 0): 4, (1, 0): 4, (2, 0): 4, (3, 0): 4, (4, 0): 4, (5, 0): 5, (0, 1): 3, (1, 1): 3, (2, 1): 3, (3, 1): 3, (4, 1): 4, (5, 1): 5, (0, 2): 2, (1, 2): 2, (


Goal has found
[(3, 1), (3, 2), (2, 1), (4, 0), (4, 2), (2, 2), (2, 0), (3, 3), (4, 1), (4, 3), (2, 3), (1, 1), (3, 0), (1, 2), (1, 0), (5, 0), (5, 1), (5, 3), (1, 3), (3,
Time taken to find the goal:  29 minutes


The final path :
dict_values([(0, 4), (1, 3), (2, 2)])
```

**TASK 05**

Random Maze 1:

[0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 8, 0]
[0, 7, 0, 0, 0, 0]
[0, 1, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0]

For DFS Search algorithm:
**Time complexity**: 32 minutes
**Completeness**: Since goal is found its complete.
**Optimality**: Compared to A start search algorithm, DFS search is not optimal since time taken is higher in DFS search to reach the goal.

For A Star Search algorithm:
**Time complexity**: 22 minutes
**Completeness**: Since goal is found its complete.
**Optimality**: Compared to DFS start search algorithm, A star search is optimal since time taken is lesser in A star search to reach the goal.

Random Maze 2:

[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[7, 0, 1, 0, 0, 8]

For DFS Search algorithm:
**Time complexity**: 30
**Completeness**: Since goal is found its complete.
**Optimality**: Compared to A start search algorithm, DFS search is not optimal since time taken is higher in DFS search to reach the goal.

For A Star Search algorithm:
**Time complexity**: 24
**Completeness**: Since goal is found its complete.
**Optimality**: Compared to DFS start search algorithm, A star search is optimal since time taken is lesser in A star search to reach the goal.

Random Maze 3:

[0, 7, 0, 1, 0, 1]
[0, 0, 0, 0, 8, 0]
[0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1]

For DFS Search algorithm:
**Time complexity**: 32
**Completeness**: Since goal is found its complete.
**Optimality**: Compared to A start search algorithm, DFS search is not optimal since time taken is higher in DFS search to reach the goal.

For A Star Search algorithm:
**Time complexity**: 12
**Completeness**: Since goal is found its complete.
**Optimality**: Compared to DFS start search algorithm, A star search is optimal since time taken is lesser in A star search to reach the goal.