Gaotong Wu
A13809639
ECE 172 HW1

Academic Integrity Policy: Integrity of scholarship is essential for an academic community. The University expects that both faculty and students will honor this principle and in so doing protect the validity of University intellectual work. For students, this means that all academic work will be done by the individual to whom it is assigned, without unauthorized aid of any kind.
By including this in my report, I agree to abide by the Academic Integrity Policy mentioned above.

Problem 1

(i)

```
>> A=[81 -10 16 14 66;-91 28 97 -42 4;13 55 -96 92 -85;91 -96 49 79 93;-63 96 80 96 -68]

A =

    81   -10    16    14    66
   -91    28    97   -42     4
    13    55   -96    92   -85
    91   -96    49    79    93
   -63    96    80    96   -68

>> B=[0 1 0 1 0;1 0 1 0 1;0 0 1 1 0;1 1 0 0 1;0 0 1 1 0]

B =

     0     1     0     1     0
     1     0     1     0     1
     0     0     1     1     0
     1     1     0     0     1
     0     0     1     1     0
```

(ii)

```
row =

     2
     4
     3
     3


col =

     1
     2
     3
     5
```

(iii)

```
>> C=A.*B

C =

     0   -10     0    14     0
   -91     0    97     0     4
     0     0   -96    92     0
    91   -96     0     0    93
     0     0    80    96     0
```

(iv)

```
>> dot(C(3,:),C(5,:))

ans =

        1152
```

(v)

```
>> max(C(:,4))

ans =

    96
```

```
>> [row,col] = find(C==max(C(:,4)))

row =

     5


col =

     4
```

(vi)

```
>> D=C.*C(1,:)

D =

           0         100           0         196           0
           0           0           0           0           0
           0           0           0        1288           0
           0         960           0           0           0
           0           0           0        1344           0
```

(vii)

```
>> dot(D(:,3),D(5,:))

ans =

     0
```

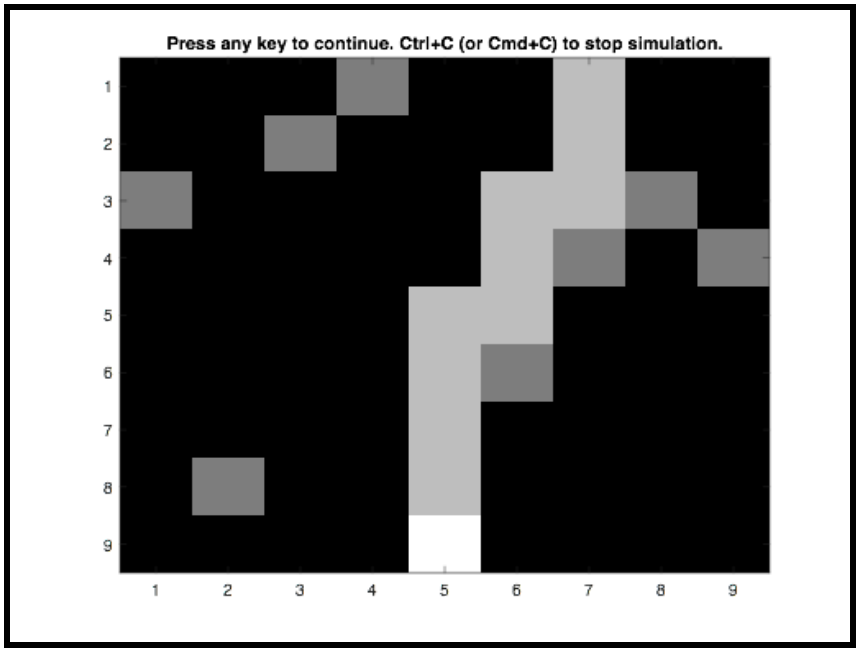
Problem 2

(i)

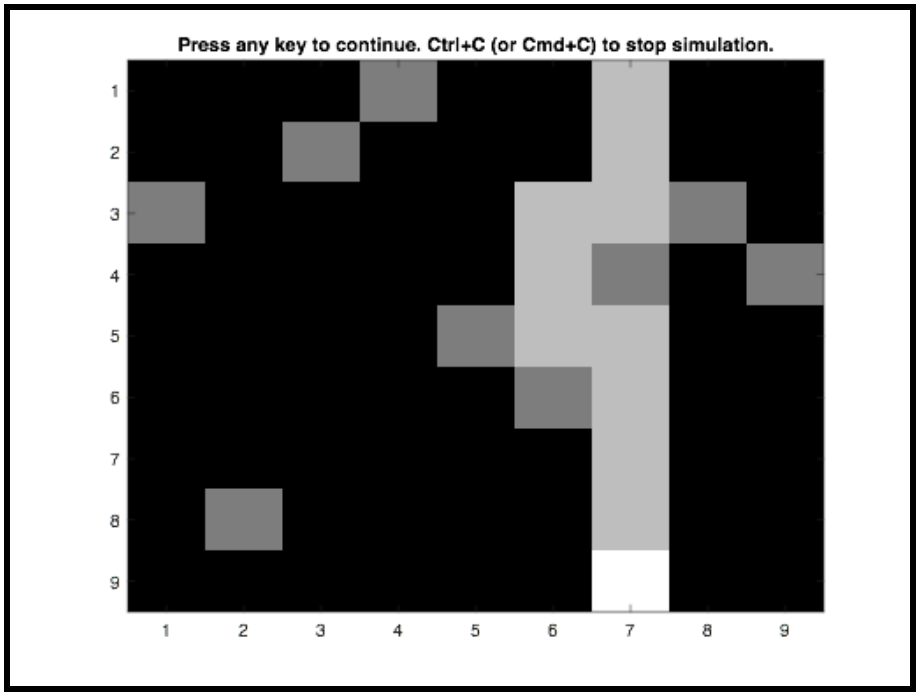
- The **loc** keeps track of the location of the bot.
- The bot will move downwards by adding line 14.
- Add an object at [4,7], the bot will collide the object and stop, because there are not commands telling the bot to go other directions (right,left,up) to keep away from the block.

- The bot can not be considered as intelligent system in that it can only move in one direction and if there are some blocks along that path it is not able to get away from them and reach the destination.

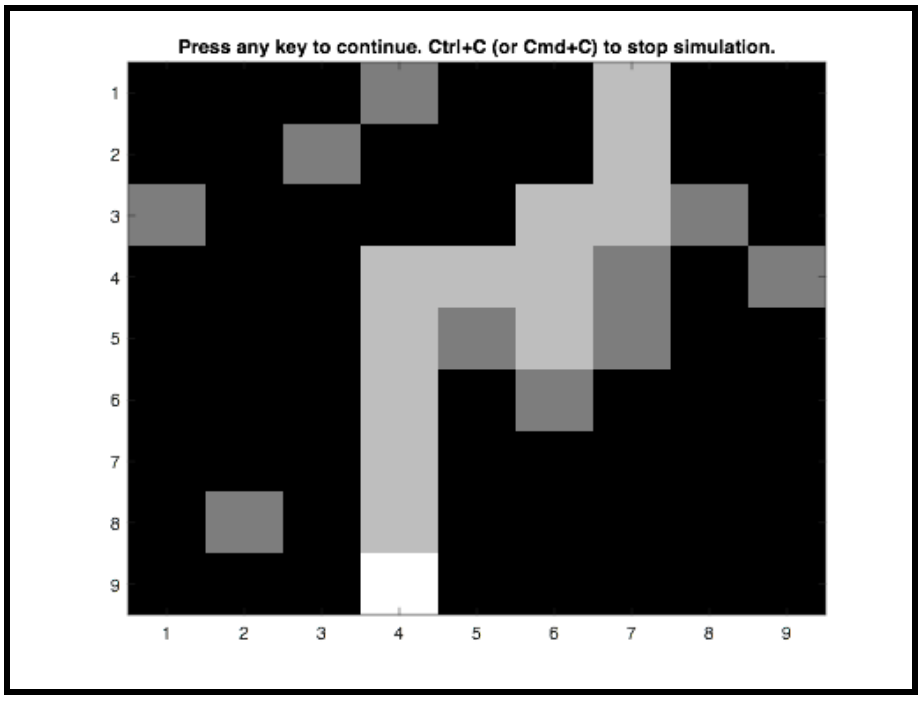
(ii)



(iii)

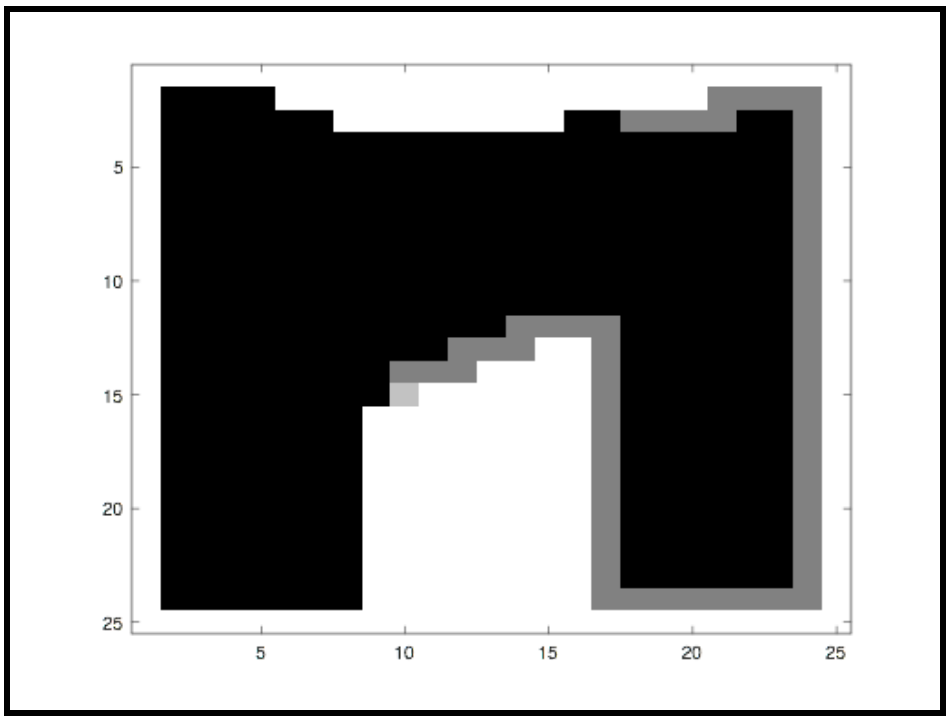

Press any key to continue. Ctrl+C (or Cmd+C) to stop simulation.
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8
9

(iv)

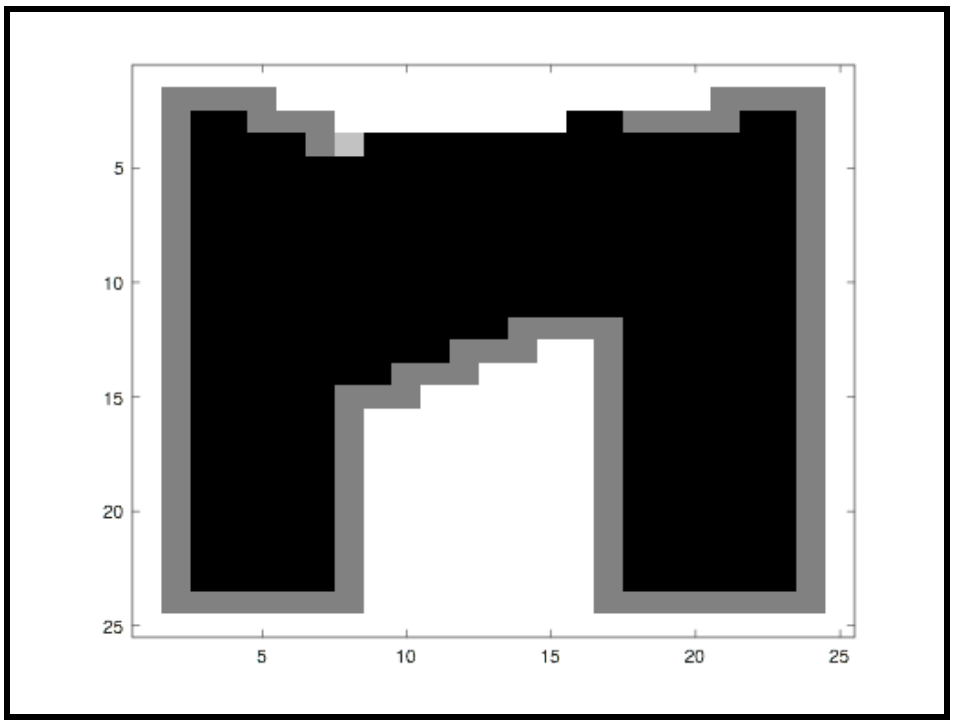

Press any key to continue. Ctrl+C (or Cmd+C) to stop simulation.
1
2
3
4
5
6
7
8
9
1
2
3
4
5
6
7
8
9

Problem 3

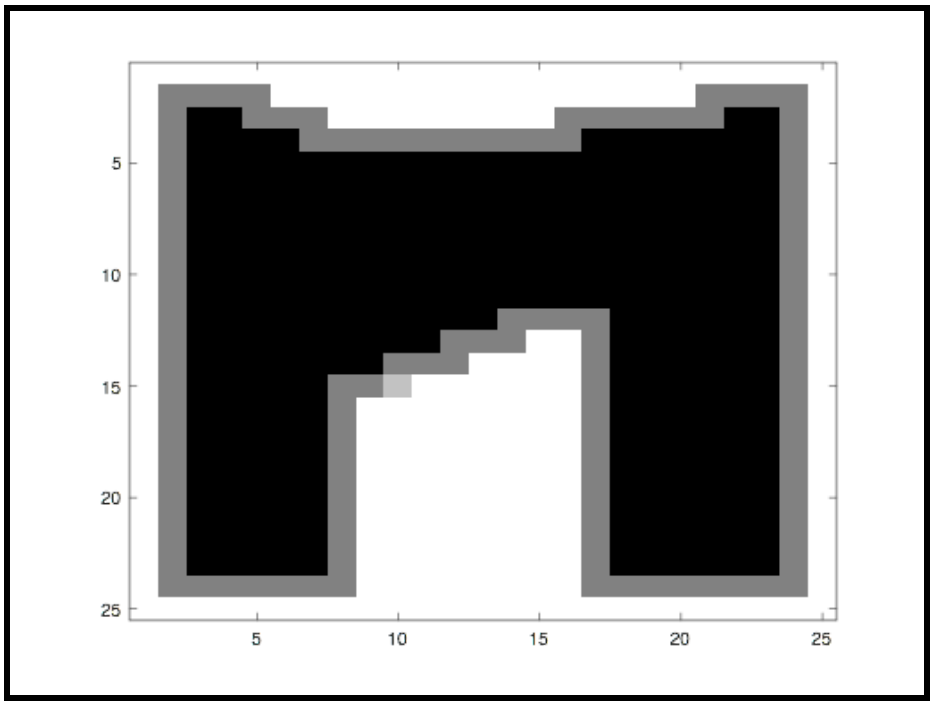
First 50%, clockwise path



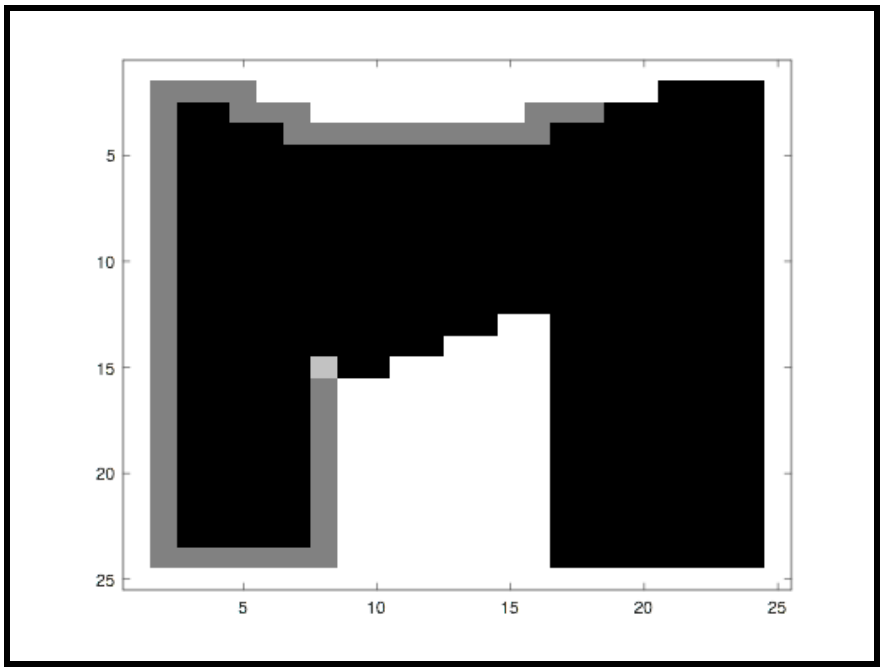
First 90%, clockwise path



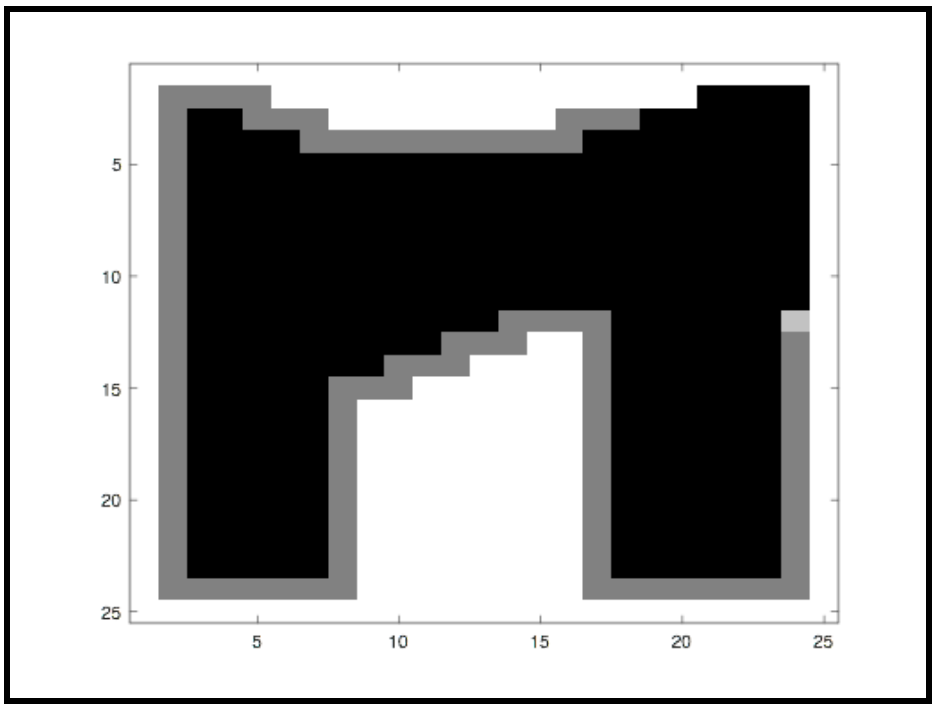
Second 50%, clockwise path



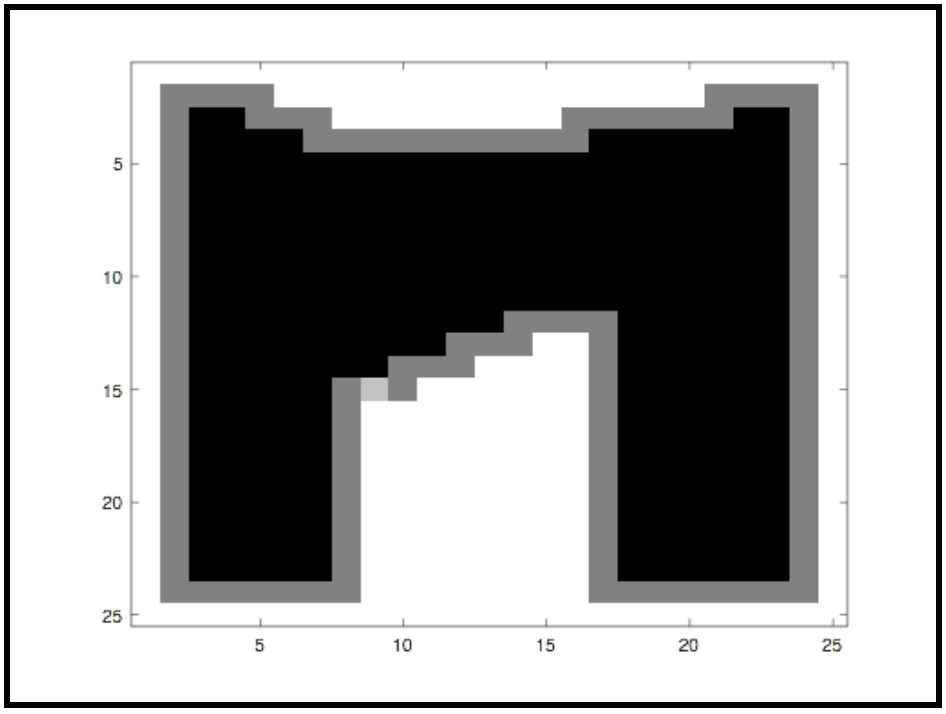
First 50%, anti-clockwise path



First 90%, anti-clockwise path



Second 50%, anti-clockwise path



Problem 5

Artificial intelligence is an outstanding invention in the 21st century and it has drastically revolutionized our lives. During June 2018, Jingdong, one of the biggest China e-commerce company, released more than 20 package delivery robots onto the streets of Haidian District in Beijing. This kind of robots is designed to deliver customers' packages as a substitute for postmen. It can be categorized as an autonomous, sensor-based robot.

The autonomous vehicles navigate via 360-degree environmental monitoring supported by radar and external sensors which stop them from colliding pedestrians, road barriers and vehicles. They're even able to recognize traffic lights and react corresponsive to them. In addition to laser radar and GPS positioning, it is equipped with a panoramic vision monitoring system, front and rear anti-collision system and ultrasonic sensing system, so that the distribution robot can accurately sense the surrounding environment changes and guarantee traffic safety. All these are done by feedback controller; it gets input from the surrounding environment and gives feedback to make direction and route changes. Besides, it is a social robot: Once the vehicle has reached its destination, customers will receive an instant message to pick up their delivery. They can do so by facial recognition, passwords or simply via JD.com's app.

By using the delivery robots, no more human is needed to deliver the online orders. However, there will be some technical problems to widespread this kind of robot. First, since it's a replacement of the human workforce to deliver packages, it might lead to a large number of unemployment, which irritates most workers. Second, is it really more economical to use these robots than hiring people? Third, the biggest robots can carry up to 30 packages; however, a regular delivery truck (eg. UPS, FedEx,) can load more than 30 packages. Unless the robot increases its capacity, otherwise it won't be competitive compared to human delivery. Last but not least, I do not find any ethical problems in using this kind of robots as they are merely used to deliver packages.

Appendix

Problem 1

```
%input matrix A and B
A=[81 -10 16 14 66;-91 28 97 -42 4;13 55 -96 92 -85;91 -96 49 79 93;-63 96
80 96 -68];
B=[0 1 0 1 0;1 0 1 0 1;0 0 1 1 0;1 1 0 0 1;0 0 1 1 0];

%Find the row and col in A that less than -70
[row,col]=find(A<-70)

%Pointwise Multiply A with B and set it to C
C=A.*B

%inner product of the 3rd and 5th row of C
dot(C(3,:),C(5,:))

%find the Max of col 4 of C and its index
max(C(:,4))
[row,col]=find(C=max(C(:,4)))

%pointwise multiply the first row of C to all its row
D=C.*C(1,:)

%(vii)
dot(D(:,3),D(5,:))
```

Problem 2

```
 % START
      out=haveIBeenHereBefore(loc,nextStep);
    if out==1
        nextStep = loc(end,:) + [0 -1];
    end
     if detectObject(loc, obj, 'S')
         nextStep = loc(end,:) + [0 -1];
     end
     if detectObject(loc, obj, 'W') && detectObject(loc, obj, 'S')
         nextStep = loc(end,:) + [0 1];
     end
     if detectObject(loc, obj, 'W') && detectObject(loc, obj, 'S') &&
detectObject(loc, obj, 'E')
         nextStep = loc(end,:) + [-1 0];
```

```
    end
  %STOP
```

Problem 3

```
if dir == 0
        if sensorInput(1,2) == 1 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0
            newPos(2) = newPos(2) + 1; % Move Right
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(1,1) == 1
            newPos(1) = newPos(1) -1; % Move Up
        end
        if sensorInput(1,2) == 1 && sensorInput(2,1) == 1 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0
            newPos(2) = newPos(2) +1; % Move Right
        end
        if sensorInput(1,2) == 1 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 1 && sensorInput(3,2) == 0
            newPos(1) = newPos(1) +1; % Move Down
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 1 && sensorInput(3,2) == 0
            newPos(1) = newPos(1) +1; % Move Down
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 1 && sensorInput(3,2) == 1
            newPos(2) = newPos(2) -1; % Move Left
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 1
            newPos(2) = newPos(2) -1; % Move Left
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 1 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 1
            newPos(1) = newPos(1) -1; % Move Up
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 1 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0
            newPos(1) = newPos(1) -1; % Move Up
        end
         if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(3,1) == 1
            newPos(2) = newPos(2) -1; % Move Left
         end
```

```
         if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(3,3) == 1
            newPos(1) = newPos(1) +1; % Move Down
         end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(1,3) == 1
            newPos(2) = newPos(2) +1; % Move Right
        end
    end

    if dir == 1
        if sensorInput(1,2) == 1 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0
            newPos(2) = newPos(2) - 1; % Move Left
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(1,1) == 1
            newPos(2) = newPos(2) -1; % Move Left
        end
        if sensorInput(1,2) == 1 && sensorInput(2,1) == 1 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0
            newPos(1) = newPos(1) +1; % Move Down
        end
        if sensorInput(1,2) == 1 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 1 && sensorInput(3,2) == 0
            newPos(2) = newPos(2) -1; % Move Left
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 1 && sensorInput(3,2) == 0
            newPos(1) = newPos(1) -1; % Move Up
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 1 && sensorInput(3,2) == 1
            newPos(1) = newPos(1) -1; % Move Up
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 1
            newPos(2) = newPos(2) +1; % Move Right
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 1 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 1
            newPos(2) = newPos(2) +1; % Move Right
        end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 1 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0
            newPos(1) = newPos(1) +1; % Move Down
        end
         if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(3,1) == 1
```

```
            newPos(1) = newPos(1) +1; % Move Down
         end
         if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(3,3) == 1
            newPos(2) = newPos(2) +1; % Move Right
         end
        if sensorInput(1,2) == 0 && sensorInput(2,1) == 0 &&
sensorInput(2,3) == 0 && sensorInput(3,2) == 0 && sensorInput(1,3) == 1
            newPos(1) = newPos(1) -1; % Move Up
        end
    end
```

Problem 4

```
% get_unexplored_area
[row,col]=find(explore_map==UNMAPPED)
  unexplored_areas=[row,col];

%get_new_destination
 N=size(unexplored_areas);
for i=1:N(1)
    d(i)=sqrt((curPos(1,1)-unexplored_areas(i,1))^2 + (curPos(1,2)-
unexplored_areas(i,2))^2);
end
[M,I]=min(d);
dest=[unexplored_areas(I,1) unexplored_areas(I,2)];

%update_explored_map
N=size(route);
for i=1:N(1)
    if explore_map(route(i,1),route(i,2))==UNMAPPED
        explore_map(route(i,1),route(i,2))=PLANNED;
    end
end
explore_map(dest(1,1),dest(1,2))=PLANNED;

%update_position
curPos=route(1,:);
explore_map(route(1,1),route(1,2))=MAPPED;
if curPos==dest
    dest = [];
end
route(1,:)=[];
```