Samyak Jain and Vedansh Mittal
Roll number = 2019101013
X = 1 - (1013)%30 / 100 = 1 - 0.24 = 0.76
Y = 2019101013%90 + 10 = 63 + 10 = 73

Initial Calculations done for calculating transition probabilities

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)

My notation:

A	B	C	D
E	F	G	H

Target Transition Probability		
Initial Pos	**Final Pos**	**Probability**
A	B	0.1
A	E	0.1
A	A	0.8
E	A	0.1
E	F	0.1
E	E	0.8
B	A	0.1
B	F	0.1
B	C	0.1
B	B	0.7
F	E	0.1
F	B	0.1
F	G	0.1
F	F	0.7
C	C	0.7
C	B	0.1
C	G	0.1
C	D	0.1
G	G	0.7
G	C	0.1
G	F	0.1
G	H	0.1

D	D	0.8
D	C	0.1
D	H	0.1
H	D	0.1
H	G	0.1
H	H	0.8

Agent Transition Probability			
Initial Pos	**Final Pos**	**Action**	**Probability**
A	B	Right	0.76
A	E	Down	0.76
A	A	Left	0.76
A	A	Up	0.76
A	A	Right	0.24
A	A	Down	0.24
A	B	Left	0.24
A	E	Up	0.24
A	A	Stay	1
E	A	Up	0.76
E	F	Right	0.76
E	E	Down	0.76
E	E	Left	0.76
E	E	Up	0.24
E	E	Right	0.24
E	A	Down	0.24
E	F	Left	0.24
E	E	Stay	1
B	A	Left	0.76
B	F	Down	0.76
B	C	Right	0.76
B	B	Up	0.76
B	C	Left	0.24
B	B	Down	0.24
B	A	Right	0.24
B	F	Up	0.24
B	B	Stay	1

F	E	Left	0.76
F	B	Up	0.76
F	G	Right	0.76
F	F	Down	0.76
F	G	Left	0.24
F	F	Up	0.24
F	E	Right	0.24
F	B	Down	0.24
F	F	Stay	1
C	D	Right	0.76
C	B	Left	0.76
C	G	Down	0.76
C	C	Up	0.76
C	B	Right	0.24
C	D	Left	0.24
C	C	Down	0.24
C	G	Up	0.24
C	C	Stay	1
G	H	Right	0.76
G	C	Up	0.76
G	F	Left	0.76
G	G	Down	0.76
G	F	Right	0.24
G	G	Up	0.24
G	H	Left	0.24
G	C	Down	0.24
G	G	Stay	1
D	H	Down	0.76
D	C	Left	0.76
D	D	Up	0.76
D	D	Right	0.76
D	D	Down	0.24
D	D	Left	0.24
D	H	Up	0.24
D	C	Right	0.24
D	D	Stay	1
H	D	Up	0.76

H	G	Left	0.76
H	H	Down	0.76
H	H	Right	0.76
H	H	Up	0.24
H	H	Left	0.24
H	D	Down	0.24
H	G	Right	0.24
H	H	Stay	1

Call			
Positions - Ag/Tg	**Initial Call**	**Final Call**	**Probability**
Different	0	1	0.5
Different	0	0	0.5
Different	1	0	0.1
Different	1	1	0.9
Same	1	0	1
Same	0	1	0.5
Same	0	0	0.5
Same	1	1	0

All the above tables were made in excel and exported as csv. They were later imported in python to calculate transition probabilities.

Python Code(calculates observation probabilities, rewards, transition probabilities in pomdp format):

```
actions_dict = {
       "L": 0,
       "R": 1,
       "S": 2,
       "U": 3,
       "D": 4}

reverse_action_map = {
     0 :  "l",
     1 :  "r",
     2 :  "s",
     3 :  "u",
     4 :  "d"
}
```

```python
positions_dict = {
#      (0,0) (0,1)    (0,2)    (0,3)
# (1,0) (1,1)    (1,2)    (1,3)
   "A":0,
   "B":1,
   "C":2,
   "D":3,
   "E":4,
   "F":5,
   "G":6,
   "H":7
}

reverse_positions_dict = {
#      (0,0) (0,1)    (0,2)    (0,3)
# (1,0) (1,1)    (1,2)    (1,3)
   0:"00",
   1:"01",
   2:"02",
   3:"03",
   4:"10",
   5:"11",
   6:"12",
   7:"13"
}

import csv

num_pos = 8
num_call = 2
num_actions = 5

states = [(agent_pos, target_pos, call) for agent_pos in range(num_pos)
         for target_pos in range(num_pos)
         for call in range(num_call)]

agent_probability = [[[0
                        for initial_pos in range(num_pos)]
                        for final_pos in range(num_pos)]
                        for action in range(num_actions)]
target_probability = [[0 for initial_pos in range(num_pos)]
                        for final_pos in range(num_pos)]
transition_prob = [[[0 for initial_state in range(len(states))]
```

```
                        for final_state in range(len(states))]
                        for action in range(num_actions)]

# transition_prob = {}
# for target_pos, agent_pos, call in states:
#     for target_pos2, agent_pos2, call2 in states:
#         for action in range(num_actions):
#             transition_prob[str(tuple([target_pos, agent_pos,
call]))][str(tuple([target_pos2, agent_pos2, call2]))][action] = 0

with open('pomdp - Sheet2.csv') as csv_file:
   csv_reader = csv.reader(csv_file, delimiter=',')
   for row in csv_reader:

target_probability[positions_dict[row[0]]][positions_dict[row[1]]] =
float(row[2])

with open('pomdp - Sheet3.csv') as csv_file:
   csv_reader = csv.reader(csv_file, delimiter=',')
   # count = 0
   for row in csv_reader:
       # print(str(count) + " " + str(row[0]) + " " + str(row[1]) + " "
+ str(row[2]))
       # print(str(count) + " " + str(actions_dict[row[0]]) + " " +
str(positions_dict[row[1]]) + " " + str(positions_dict[row[2]]))

agent_probability[actions_dict[row[0]]][positions_dict[row[1]]][positio
ns_dict[row[2]]] = float(row[3])
       # count += 1

with open('transition_probabilities.txt', 'w+') as f:
   for action in range(num_actions):
       for initial_state in range(len(states)):
           for final_state in range(len(states)):
               initial_agent_pos = states[initial_state][0]
               initial_target_pos = states[initial_state][1]
               initial_call = states[initial_state][2]

               final_agent_pos = states[final_state][0]
               final_target_pos = states[final_state][1]
               final_call = states[final_state][2]

               if(initial_target_pos==initial_agent_pos):
```

```
                if(initial_call==0 and final_call==0):
                    call_probability = 0.5
                elif(initial_call==0 and final_call==1):
                    call_probability = 0.5
                elif(initial_call==1 and final_call==0):
                    call_probability = 1
                elif(initial_call==1 and final_call==1):
                    call_probability = 0

            if(initial_target_pos!=initial_agent_pos):
                if(initial_call==0 and final_call==0):
                    call_probability = 0.5
                elif(initial_call==0 and final_call==1):
                    call_probability = 0.5
                elif(initial_call==1 and final_call==0):
                    call_probability = 0.1
                elif(initial_call==1 and final_call==1):
                    call_probability = 0.9

            transition_prob[action][initial_state][final_state] =
agent_probability[action][initial_agent_pos][final_agent_pos]*target_pr
obability[initial_target_pos][final_target_pos]*call_probability


if(transition_prob[action][initial_state][final_state]!=0):
                f.write("T: " + str(reverse_action_map[action]) + "
: S" + str(reverse_positions_dict[initial_agent_pos]) +
str(reverse_positions_dict[initial_target_pos]) + str(initial_call) + "
: S" + str(reverse_positions_dict[final_agent_pos]) +
str(reverse_positions_dict[final_target_pos]) + str(final_call) + " " +
str(transition_prob[action][initial_state][final_state]) + '\n')

with open('observations.txt', 'w+') as f:
   for final_state in range(len(states)):

       final_agent_pos = states[final_state][0]
       final_target_pos = states[final_state][1]
       final_call = states[final_state][2]

       if(final_target_pos==final_agent_pos):
           num = 1
       elif((final_target_pos==0 and final_agent_pos==1 )
       or (final_target_pos==1 and final_agent_pos==2 )
```

```
        or (final_target_pos==2 and final_agent_pos==3 )
        or (final_target_pos==4 and final_agent_pos==5 )
        or (final_target_pos==5 and final_agent_pos==6 )
        or (final_target_pos==6 and final_agent_pos==7 )):
            num = 4
        elif((final_target_pos==1 and final_agent_pos==0 )
        or (final_target_pos==2 and final_agent_pos==1 )
        or (final_target_pos==3 and final_agent_pos==2 )
        or (final_target_pos==5 and final_agent_pos==4 )
        or (final_target_pos==6 and final_agent_pos==5 )
        or (final_target_pos==7 and final_agent_pos==6 )):
            num = 2
        elif((final_target_pos==0 and final_agent_pos==4 )
        or (final_target_pos==1 and final_agent_pos==5 )
        or (final_target_pos==2 and final_agent_pos==6 )
        or (final_target_pos==3 and final_agent_pos==7 )):
            num = 5
        elif((final_target_pos==4 and final_agent_pos==0 )
        or (final_target_pos==5 and final_agent_pos==1 )
        or (final_target_pos==6 and final_agent_pos==2 )
        or (final_target_pos==7 and final_agent_pos==3 )):
            num = 3
        else:
            num = 6

        f.write("O :" + " * : "+ "S" +
str(reverse_positions_dict[final_agent_pos]) +
str(reverse_positions_dict[final_target_pos]) + str(final_call) + " : "
+ "o" + str(num) + " 1.0" + '\n')

with open('rewards.txt', 'w+') as f:
   for action in range(num_actions):
        for final_state in range(len(states)):

            final_agent_pos = states[final_state][0]
            final_target_pos = states[final_state][1]
            final_call = states[final_state][2]
            if(action==2):
                rew = 0
            else:
                rew = -1

            if((final_target_pos==final_agent_pos) and (final_call==1)):
```

```
                num = 1
                rew = 73 + rew
            elif(final_target_pos==final_agent_pos) and (final_call==0):
                num = 1
            elif((final_target_pos==0 and final_agent_pos==1 )
            or (final_target_pos==1 and final_agent_pos==2 )
            or (final_target_pos==2 and final_agent_pos==3 )
            or (final_target_pos==4 and final_agent_pos==5 )
            or (final_target_pos==5 and final_agent_pos==6 )
            or (final_target_pos==6 and final_agent_pos==7 )):
                num = 4
            elif((final_target_pos==1 and final_agent_pos==0 )
            or (final_target_pos==2 and final_agent_pos==1 )
            or (final_target_pos==3 and final_agent_pos==2 )
            or (final_target_pos==5 and final_agent_pos==4 )
            or (final_target_pos==6 and final_agent_pos==5 )
            or (final_target_pos==7 and final_agent_pos==6 )):
                num = 2
            elif((final_target_pos==0 and final_agent_pos==4 )
            or (final_target_pos==1 and final_agent_pos==5 )
            or (final_target_pos==2 and final_agent_pos==6 )
            or (final_target_pos==3 and final_agent_pos==7 )):
                num = 5
            elif((final_target_pos==4 and final_agent_pos==0 )
            or (final_target_pos==5 and final_agent_pos==1 )
            or (final_target_pos==6 and final_agent_pos==2 )
            or (final_target_pos==7 and final_agent_pos==3 )):
                num = 3
            else:
                num = 6

            # for i in range(1,7):
            #     if(i==1):
            #         if(final_target_pos==final_agent_pos):
            #             rew = 72
            #         else:
            #             rew = -1
            #     else:
            #         rew = -1
            f.write("R: " + str(reverse_action_map[action]) + " : * : S"
+ str(reverse_positions_dict[final_agent_pos]) +
str(reverse_positions_dict[final_target_pos]) + str(final_call) + " :
o" + str(num) + " " + str(rew) + '\n')
```

```
def q1_initial_belief_state():
   with open('q1_initial_belief_state.txt', 'w+') as f:
       for state in range(len(states)):

           agent_pos = states[state][0]
           target_pos = states[state][1]

           if(target_pos==4):
               if(agent_pos in (1,2,3,6,7)):
                   f.write("0.1 ")
               else:
                   f.write("0.0 ")
           else:
               f.write("0.0 ")

   with open('q1_initial_belief_state.txt', 'a') as f:
       f.write('\n')
       for i in range(128):
           if(i<10):
               f.write(str(i) + "   ")
           elif(i>=10 and i<=99):
               f.write(str(i) + "  ")
           else:
               f.write(str(i) + " ")

def q2_initial_belief_state():
   with open('q2_initial_belief_state.txt', 'w+') as f:
       for state in range(len(states)):

           agent_pos = states[state][0]
           target_pos = states[state][1]
           call = states[state][2]

           if(call==1):
               if(agent_pos==5):
                   if(target_pos in (1,4,5,6)):
                       f.write("0.25 ")
                   else:
                       f.write("0.0 ")
               else:
                   f.write("0.0 ")
           else:
```

```
                f.write("0.0 ")

def state_mapping():
    with open('state_map.txt', 'w+') as f:
        i = 0
        for state in range(len(states)):
            agent_pos = states[state][0]
            target_pos = states[state][1]
            call = states[state][2]

            f.write("S" + str(reverse_positions_dict[agent_pos]) +
str(reverse_positions_dict[target_pos]) + str(call) + " ")
            # f.write(str(i))
            i += 1

q1_initial_belief_state()
q2_initial_belief_state()
state_mapping()
```

Explanation:
https://github.com/AdaCompNUS/sarsop/blob/master/doc/POMDP/PomdpFileFormat.html
This file format was followed

```
R: <action> : <start-state> : <end-state> :
<observation> %f
T: <action> : <start-state> : <end-state> %f
O : <action> : <end-state> : <observation> %f
```

State is dependent on target position, agent position and call value i.e 8*8*2 = 128 states in total

Nomenclature Used:

1. States are represented as S01011 where first two digits indicate agent position coordinates, next two digits indicate target position coordinates and the final digit indicates the call value. This helps a lot in debugging.

Q1

Initial Belief State:

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0

Mapping between state and probability:

S00000 0.0
S00001 0.0
S00010 0.0
S00011 0.0
S00020 0.0
S00021 0.0
S00030 0.0
S00031 0.0
S00100 0.0
S00101 0.0
S00110 0.0
S00111 0.0
S00120 0.0
S00121 0.0
S00130 0.0
S00131 0.0
S01000 0.0
S01001 0.0
S01010 0.0
S01011 0.0
S01020 0.0
S01021 0.0
S01030 0.0
S01031 0.0
S01100 0.1
S01101 0.1
S01110 0.0
S01111 0.0
S01120 0.0
S01121 0.0
S01130 0.0
S01131 0.0
S02000 0.0
S02001 0.0
S02010 0.0
S02011 0.0
S02020 0.0
S02021 0.0
S02030 0.0
S02031 0.0
S02100 0.1
S02101 0.1
S02110 0.0
S02111 0.0
S02120 0.0
S02121 0.0
S02130 0.0
S02131 0.0

S03000 0.0
S03001 0.0
S03010 0.0
S03011 0.0
S03020 0.0
S03021 0.0
S03030 0.0
S03031 0.0
S03100 0.1
S03101 0.1
S03110 0.0
S03111 0.0
S03120 0.0
S03121 0.0
S03130 0.0
S03131 0.0
S10000 0.0
S10001 0.0
S10010 0.0
S10011 0.0
S10020 0.0
S10021 0.0
S10030 0.0
S10031 0.0
S10100 0.0
S10101 0.0
S10110 0.0
S10111 0.0
S10120 0.0
S10121 0.0
S10130 0.0
S10131 0.0
S11000 0.0
S11001 0.0
S11010 0.0
S11011 0.0
S11020 0.0
S11021 0.0
S11030 0.0
S11031 0.0
S11100 0.0
S11101 0.0
S11110 0.0
S11111 0.0
S11120 0.0
S11121 0.0
S11130 0.0
S11131 0.0

S12000 0.0
S12001 0.0
S12010 0.0
S12011 0.0
S12020 0.0
S12021 0.0
S12030 0.0
S12031 0.0
S12100 0.1
S12101 0.1
S12110 0.0
S12111 0.0
S12120 0.0
S12121 0.0
S12130 0.0
S12131 0.0
S13000 0.0
S13001 0.0
S13010 0.0
S13011 0.0
S13020 0.0
S13021 0.0
S13030 0.0
S13031 0.0
S13100 0.1
S13101 0.1
S13110 0.0
S13111 0.0
S13120 0.0
S13121 0.0
S13130 0.0
S13131 0.0

Initial Belief State calculation:
Target is on E and observes o6 which means agent is on B,C,D,G,H with equal probability. Also call can take 2 values so assign 0.1 probability to these 10 states corresponding to agent position and call value.

Transition probability can be calculated by multiplying respective agent, target and call probabilities for each state transition.

Observation Probabilities are 1.0. There is a single observation per end state which is independent of action. It depends on the relative position of agent and target.

Reward is dependent on whether the agent position and target position are the same or not. It is also dependent on the final call value. It is also dependent on action as for STAY action step cost is 0 and is -1 for other actions. It is independent of the start state.

Run this command to get policy:
./pomdpsol ../../q1.pomdp

Run this command to get expected reward for a particular number of simulations:
./pomdpsim --simLen 100 --simNum 1000 --policy-file out.policy ../../q1.pomdp

Q2
Initial belief state:
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.25 0.0 0.0 0.0 0.0 0.0 0.25 0.0 0.25 0.0 0.25 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Mapping between state and probability:
S00000 0.0
S00001 0.0
S00010 0.0
S00011 0.0
S00020 0.0
S00021 0.0
S00030 0.0
S00031 0.0
S00100 0.0
S00101 0.0
S00110 0.0
S00111 0.0
S00120 0.0
S00121 0.0
S00130 0.0
S00131 0.0
S01000 0.0
S01001 0.0
S01010 0.0
S01011 0.0
S01020 0.0
S01021 0.0
S01030 0.0
S01031 0.0
S01100 0.0
S01101 0.0
S01110 0.0
S01111 0.0
S01120 0.0
S01121 0.0
S01130 0.0
S01131 0.0
S02000 0.0

S02001 0.0
S02010 0.0
S02011 0.0
S02020 0.0
S02021 0.0
S02030 0.0
S02031 0.0
S02100 0.0
S02101 0.0
S02110 0.0
S02111 0.0
S02120 0.0
S02121 0.0
S02130 0.0
S02131 0.0
S03000 0.0
S03001 0.0
S03010 0.0
S03011 0.0
S03020 0.0
S03021 0.0
S03030 0.0
S03031 0.0
S03100 0.0
S03101 0.0
S03110 0.0
S03111 0.0
S03120 0.0
S03121 0.0
S03130 0.0
S03131 0.0
S10000 0.0
S10001 0.0
S10010 0.0
S10011 0.0
S10020 0.0
S10021 0.0
S10030 0.0
S10031 0.0
S10100 0.0
S10101 0.0
S10110 0.0
S10111 0.0
S10120 0.0
S10121 0.0
S10130 0.0
S10131 0.0
S11000 0.0

S11001 0.0
S11010 0.25
S11011 0.0
S11020 0.0
S11021 0.0
S11030 0.0
S11031 0.0
S11100 0.25
S11101 0.0
S11110 0.25
S11111 0.0
S11120 0.25
S11121 0.0
S11130 0.0
S11131 0.0
S12000 0.0
S12001 0.0
S12010 0.0
S12011 0.0
S12020 0.0
S12021 0.0
S12030 0.0
S12031 0.0
S12100 0.0
S12101 0.0
S12110 0.0
S12111 0.0
S12120 0.0
S12121 0.0
S12130 0.0
S12131 0.0
S13000 0.0
S13001 0.0
S13010 0.0
S13011 0.0
S13020 0.0
S13021 0.0
S13030 0.0
S13031 0.0
S13100 0.0
S13101 0.0
S13110 0.0
S13111 0.0
S13120 0.0
S13121 0.0
S13130 0.0
S13131 0.0

There is 0.25 probability for state with agent position F and target position B,E,F,G and call value 1. Other values remain the same from the previous question.

Q3

Expected Total Reward (I have given screenshots after running for 4 times)

For q1

12.3283

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              12.3283            (11.7203, 12.9363)
-------------------------------------------------------------
```

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              13.2313            (12.6234, 13.8392)
-------------------------------------------------------------
```

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              12.2102            (11.6096, 12.8108)
-------------------------------------------------------------
```

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              12.578             (11.9772, 13.1788)
-------------------------------------------------------------
```

For q2

25.1957

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              25.1957            (24.6926, 25.6989)
-------------------------------------------------------------
```

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              26.0903            (25.6174, 26.5632)
-------------------------------------------------------------
```

```
-------------------------------------------------------------
#Simulations    | Exp Total Reward | 95% Confidence Interval
-------------------------------------------------------------
1000              26.2319            (25.7504, 26.7134)
-------------------------------------------------------------
```

```
---------------------------------------------------------------
#Simulations  | Exp Total Reward | 95% Confidence Interval
---------------------------------------------------------------
1000            25.6764             (25.1912, 26.1615)
---------------------------------------------------------------
```

Q4

Probabilities mentioned for target and agent

A - 0.4	T - 0.25	T - 0.25	
	T - 0.25	T - 0.25	A - 0.6

Calculate the probability for each observation.

Take all the 8 cases

Agent Position	Target Position	Probability	Observation
A	B	0.4*0.25 = 0.1	o2
A	C	0.4*0.25 = 0.1	o6
A	F	0.4*0.25 = 0.1	o6
A	G	0.4*0.25 = 0.1	o6
H	B	0.6*0.25 = 0.15	o6
H	C	0.6*0.25 = 0.15	o6
H	F	0.6*0.25 = 0.15	o6
H	G	0.6*0.25 = 0.15	o4

Final Probabilities:

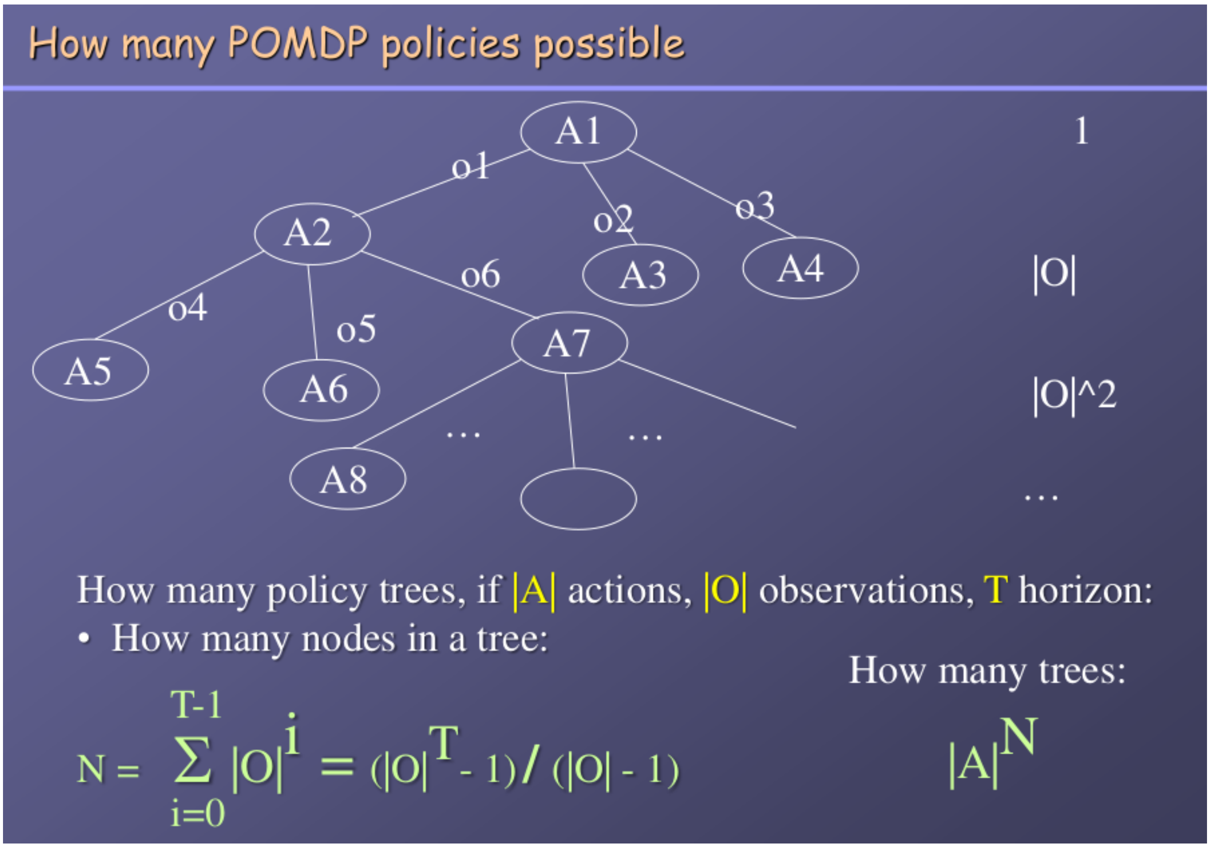
O2 -> 0.1

O6 -> 0.75

O4 -> 0.15

Agent will most likely sense the target to be not in its 1 cell neighbourhood as o6 has the highest probability.

Q5



$|O| = 6$
$|A| = 5$
T = variable not specified

$N = 6^T - 1 / 5$

No. of policy trees = A^N = 5 ^ (6^T - 1 / 5)