EXPERIMENT - 7

IMPLEMENTATION OF UNCERTAIN METHODS FOR AN APPLICATION

AIM

To implement Monty Hall Problem using CPP programming.

ALGORITHM

1 The main goal by the puzzle is to maximise the chances to win the game in the beginning the guest can choose one of three doors. The player can keep the same door or switch to other door The aim is to calculate the probability with and without switching

2. Suppose guest start from door one the host shows other door that does not contain the car

3. If the car is behind door 1. then after guest picks door 1 the host opens either 2 or 3 and guest switches to remaining doors and loses

4. If the car is behind door 2 then after the guest picks 1 the host switches to 3. The guest chooses 2 and wins

5. If the car is behind door 3 then guest picks door number 1, the host is forced to open door 2 then guest switches to door 3 and wins

6. Thus in 2 of 3 probabilities there are more chances of winning, chances of winning is 2/3 due to switcing

According to Baye's Theorem

$$P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right) \times P(A)}{P(B)}$$

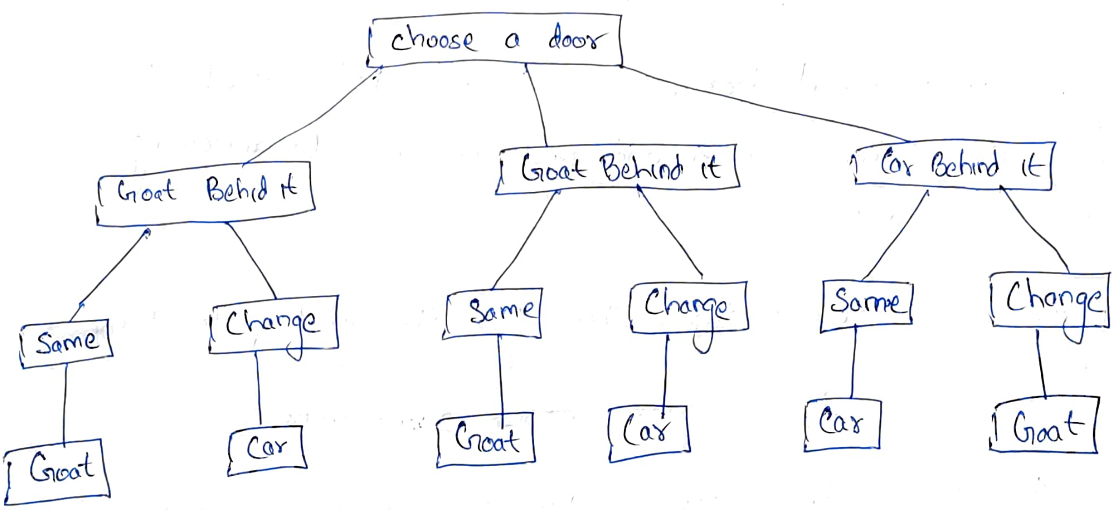
$$= \frac{\frac{1}{2} \times \frac{1}{3}}{\frac{1}{3} \times \frac{1}{2} + \frac{1}{3} \times 0 + \frac{1}{3} \times 1}$$

$$= \frac{1}{3}$$

7. The chances that is behind door 1 is $\frac{1}{3}$. As the 2 doors are left now they will have $\frac{1}{2}$ probability of getting chosen

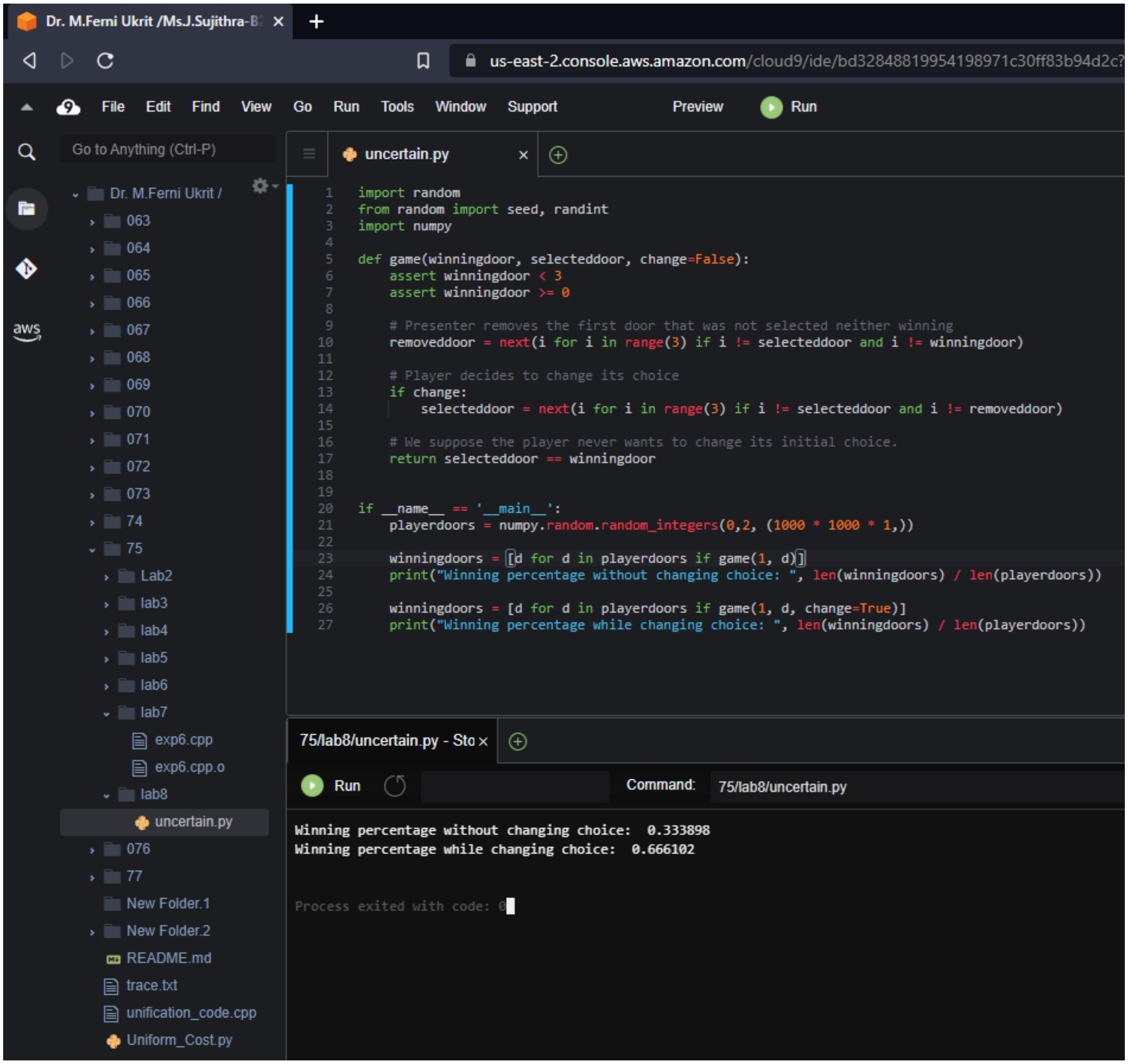
8. When guest chooses gate 1 the host shows goat behind door 2 if the car is behind door 2. host opens door 3 So the probability is $\frac{1}{3}$.

WORKING



RESULT

Monty Hall problem was executed and verified


Dr. M.Femi Ukrit /Ms.J.Sujithra-B
us-east-2.console.aws.amazon.com/cloud9/ide/bd32848819954198971c30ff83b94d2c
File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
uncertain.py
Dr. M.Femi Ukrit /
063
064
065
066
067
068
069
070
071
072
073
74
75
Lab2
lab3
lab4
lab5
lab6
lab7
exp6.cpp
exp6.cpp.o
lab8
uncertain.py
076
77
New Folder.1
New Folder.2
README.md
trace.txt
unification_code.cpp
Uniform_Cost.py
import random
from random import seed, randint
import numpy
def game(winningdoor, selecteddoor, change=False):
assert winningdoor < 3
assert winningdoor >= 0
# Presenter removes the first door that was not selected neither winning
removeddoor = next(i for i in range(3) if i != selecteddoor and i != winningdoor)
# Player decides to change its choice
if change:
selecteddoor = next(i for i in range(3) if i != selecteddoor and i != removeddoor)
# We suppose the player never wants to change its initial choice.
return selecteddoor == winningdoor
if __name__ == '__main__':
playerdoors = numpy.random.random_integers(0,2, (1000 * 1000 * 1,))
winningdoors = [d for d in playerdoors if game(1, d)]
print("Winning percentage without changing choice: ", len(winningdoors) / len(playerdoors))
winningdoors = [d for d in playerdoors if game(1, d, change=True)]
print("Winning percentage while changing choice: ", len(winningdoors) / len(playerdoors))
75/lab8/uncertain.py - Sto
Run
Command: 75/lab8/uncertain.py
Winning percentage without changing choice: 0.333898
Winning percentage while changing choice: 0.666102
Process exited with code: 0