**VIETNAM NATIONAL UNIVERSITY**

**UNIVERSITY OF SCIENCE**

**FACULTY OF INFORMATION TECHNOLOGY**



**PROJECT 01**

**ARTIFICIAL INTELLIGENCE**

VÕ VĂN QUỐC HUY 18127113

VŨ CÔNG MINH 18127000

TỪ KIẾN VINH 18127000

TỪ KIẾN HOA 18127000

**Ho Chi Minh City – 2020**

# TABLE OF CONTENTS

[TABLE OF CONTENTS 2](#_Toc48161133)

[Assignment plan and overall 3](#_Toc48161134)

[Explaning code: Object.py 4](#_Toc48161135)

[Explaning code: pacman.py 5](#_Toc48161136)

[Explaning code: Searching\_Algorithm.py 6](#_Toc48161137)

[References 7](#_Toc48161138)

# Assignment plan and overall

|  |  |  |
| --- | --- | --- |
| No. | Specification | Complete |
| 1 | Finish level 1 successfully. |  |
| 2 | Finish level 2 successfully. |  |
| 3 | Finish level 3 successfully. |  |
| 4 | Finish level 4 successfully. |  |
| 5 | Graphical demonstration of each step of the running process. |  |
| 6 | Generate at least 5 maps with difference in number and structure of walls, monsters, and food. |  |
| 7 | Report |  |

# Explaning code: Object.py

class pacman(object):

* Initialize pacman character from png image file.
* Draw pacman at position x, y read from the map file.

class monster(object):

class food(object):

# Explaning code: pacman.py

Function def random\_Maze():

* Randomly initialize two variables for map size, then create a random matrix with three values of 0, 1, 2 and then surround the matrix with value 1.
* Then random the number and value of Ghost position corresponding to a random position in the matrix.
* Assign matrix into array corresponding to each row of matrix and return the results.

Function def handle\_input():

* Initialize the variable lst as an array, open the file containing the position and matrix.
* For each line read, it is detached and fed into lst

Function def create\_maze(C):

* For each input box from the matrix will be considered at that position as Wall or Food or Ghost.
* For each position there will be additional functions.

Function def create\_data(C):

Function def update\_adjacent\_list(C):

Function def display\_score():

Function def RunAlgorithm():

Function def Controls():

# Explaning code: Searching\_Algorithm.py

Function def BFS(adjacency\_list, begin, food\_position):

* Create the parent array the size of the passed list.
* Check at start is Food or not, if so then return results.
* Create the queue for the starting value, node at the position taken from the queue. Check to see if it is in the list expanded or not, if the node at location is Food, retrieve it and return the function values.

Function def DFS(adjacency\_list, begin, food\_position):

Create the parent array the size of the passed list.

* Check at start is Food or not, if so then return results.
* Create a stack array and put the initial value in. The node at position will be populated from the stack and checked to see if it is in expand nodes. If not, will update the stack and parent again.
* If it is at that position, Food will trace back and return the values of the function.

# References