SCHOOL OF COMPUTING (SOC)

Diploma in Information Technology

ST1507 DATA STRUCTURES AND ALGORITHMS (AI)

2020/2021 SEMESTER 2
ASSIGNMENT ONE (CA1)
~ Pizza Runners ~

Objective of Assignment

To practice what you have learnt in the module by developing a suitable path finding algorithm to help a pizza delivery company to use drones to navigate a city landscape.

Instructions and Guidelines:

1. This is an individual assignment and it accounts to **30%** of your final grade.
2. The submission date is **Sunday 29 November 11:59 pm**.
3. The development will be carried out in Python using Anaconda.
4. The interviews will be conducted during the DSAA lessons in week 7. You are expected to explain on your code and program logic. Take note that the interview is compulsory. In case you are absent from the interview without valid reasons, you will not get any marks for the assignment.
5. No marks will be given, if the work is copied, or you have allowed others to copy your work.
6. **50% of marks** will be deducted for submission of assignment within **ONE** calendar day after the deadline. **No marks shall** be awarded for assignments submitted **more than one day** after the deadline.

Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person. Plagiarism is a serious offence and disciplinary action will be taken against you. If you are guilty of plagiarism, you may fail <u>all</u> modules in the semester, or even be liable for <u>expulsion</u>.

Overview of the system

Pizza Runners is a startup company that specializes in door-step pizza delivery for the busy office workers in the business district. Since last year they have started to use drones to deliver pizzas at the office workers doorsteps. They are currently using a computerized system that controls the navigation of the drones using a path finding approach that is based on the *left-hand* algorithm (also known as *wall follower* algorithm).

Lately they have received complaints from customers with regards to late deliveries, and occasionally, even completely failed deliveries. A preliminary investigation by the company's IT department, has concluded that the left-hand algorithm the company is currently using is at fault. They discovered that the left-hand algorithm has some serious short-comings. For instance the algorithm appears not always to be able to find a path between two points (even if there is one). This has caused the drone to get lost or trapped at several occasions. They also noticed, that even when the drone reaches its destination, it may not necessary have taken the shortest path, and this has caused the delays in delivery that people have been complaining about.

Your job as an AI programmer

As an AI programmer, you have been roped in to implement a more efficient, and effective path finding solution for Pizza Runners. Your task is to develop a Python application that would allow the company to simulate the navigation of the pizza delivery drones in the business district. You are required to implement the left-hand algorithm, as well as a second algorithm (one of you own choice) that would meet the requirements of the client, namely, an algorithm that is guaranteed to find a path (if there is one), and also to come up with the shortest path possible.

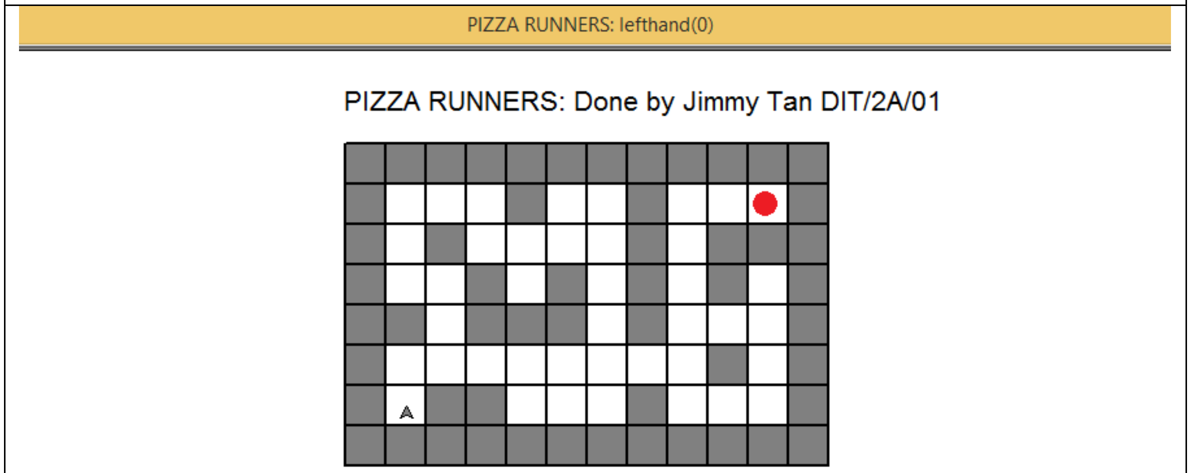
City map

You will need to read the city map from a text file. You must follow the following conventions.

Character	**Description**
X	*Building (drone can't go here)*
.	*Road (drone can fly here)*
s	*The start location of the drone*
e	*The pizza delivery location*

Next is an example of a city map file, together with a screen shot of how the city map would appear in the application.

```
XXXXXXXXXXXX
X...X..X..eX
X.X....X.XXX
X..X.X.X.X.X
XX.XXX.X...X
X........X.X
XsXX...X...X
XXXXXXXXXXXX
```


PIZZA RUNNERS: lefthand(0)
PIZZA RUNNERS: Done by Jimmy Tan DIT/2A/01

Basic requirements

- Your program should support at least two different algorithms:
  1) First you need to implement the '*left-hand rule*' algorithm.
     A left-hand algorithm, is how a blind person would use his/her left hand to follow the wall in a building to find the exit. Hence this algorithm is also referred to as '*wall follower*' algorithm (pls. do some further research on the internet)
  2) The second algorithm/approach can be any algorithm of your own choice.

- You are required to design, and write the Python application using an object oriented approach (OOP). You should use appropriate functions, classes and data structures.

- The graphics, and interactivity capabilities of the application must be implemented through the Turtle library.

- You should only make use of those libraries that already ship with Anaconda Jupyter (take note Anaconda already ships with Turtle)

- It is advisable to test your application with various city maps. Take note, we will test your application with some pre-prepared city maps.

- You must display your name and class above the city map in following format:

  PIZZA RUNNERS: Done by Your Name, Your Class

- You must display the algorithm name, and the number of steps the application took/has taken, in the window's title bar.

- It should be possible for the user to switch between algorithms with a key press. You should use the tab key for this.

- You application must at least support city maps with dimensions, of 8 rows by 12 columns (as shown in screenshot example)

Additional features:

After you have implemented the basic requirements for the drone navigation simulation application, you may implement additional features such as, but not limited to, the following:

- Ability to pause/resume the drone navigation.
- Ability to reset, and run again (without the need to restart the application).
- Ability to have multiple drones running together.
- Ability to visualize the path that the drone has followed/will be following.
- Ability for the users themselves (with key presses) to navigate the city map (with collision detection).
- Support for flexible city maps (e.g. support for dimensions other than 8 by 12)
- An extra, third type, of path finding algorithm.
- Interactive city map editor capabilities.
- City map generation capabilities.
- Any other features that may enhance the system.

(*) You may score up to a maximum of 20/100 marks for additional features.

Deliverables

Your deliverables must include:

1. A report (as pdf file) with a maximum **of 4** pages, excluding the source listing. The report should contain:

    a) Description, and user guidelines as on how to operate your application (please include screen shots of your application in action).
    b) Description, and comparison of the different path finding algorithms that you have implemented. You may discuss on issues such as, challenges faced, performance issues (e.g. Big O), data structures that you used, limitations and constraints etc.
    c) All your python source code listings, must be included as an appendix at the end of your report.

2. You may either submit the python files (.py files), or Jupyter Notebook. Make sure the code that you submit is complete, and that it can run in Anaconda (either from Anaconda Prompt or from Jupyter Notebook).

Submission deadline

Final Submission is on **Sunday 29 November 11:59 pm**

-Submit all the deliverables (Source Code, Report) in the designated Blackboard drop box before the deadline.

- You must submit it as one Zipped folder (RAR will not be accepted, only zip) whereby you label your submission as:

CA1_Name_StudentID_Class.zip

For example: *CA1_JimmyTan_12345_DIT2A01.zip*

Assessment Criteria

The assignment will be assessed based on the following criteria:

Assessment criteria	**Marks awarded**
File IO & GUI: - Reads city map correctly from file. - Displays city map correctly. - GUI allows switching of algorithms, updates algorithm name, and number of steps.	Max 15
Pathfinding algorithms: - Implement at least two algorithms, the left-hand rule algorithm and one algorithm of your own choice.	Max 20
Programming techniques and readability of code: - Usage of classes. - Usage of data structures. - Functions and programming constructs. - Comments. - Free of crashes.	Max 25
Report: - User guidelines to operate the application (including screenshots). - Description and comparison of the path finding algorithms implemented.	Max 10
Demo: - Ability to demonstrate and explain the application. - Q&A	Max 10
Extra features	Max 20
Grand Total	**100**

-- End --