CS 386 — Game Dev — Python Maze

Project #1 – Python Maze

Introduction

This project is to write a Python program to build a random maze using one of the following two algorithms as discussed in class: 1) Binary Space Partition, or 2) Kruskal Merge.

Your program should read two integers (as strings): the number of rows and columns in the maze.

Then build the maze and output it using hyphens and vertical strokes ("-", "|") for wall characters and plus signs (+) for corner characters. (This output format is best viewed with a non-proportionally spaced font.) Label the upper left cell with "S" for start and the lower right cell with "X".

Here is an example 5x5 maze output.

```
+-+-+-+-+-+
|S    |   |
+-+ +-+-+ +
|   | | | |
+ +-+ + + +
|   | |   |
+-+ + +-+ +
|         |
+ +-+-+ + +
|     | |X|
+-+-+-+-+-+
```

Hint: For each cell (AKA square) in the maze, you probably want only to print its left and top parts (characters). Then after printing the end of each row, add a vertical end wall part, and after the bottom row of cells add a full-length set of bottom wall parts.

Portals

To make it clear how your program is working, you should print out the location coordinates of each portal (AKA opening connecting two neighboring regions, or missing wall part) that your algorithm defines, in the order that these portals are defined. Each algorithm defines a sequence of such portals.

Test Runs

Include an 8x8 maze output file (with portals) and ditto for a 10x10 maze, as two test run files in your .zip file submission.

Team

Your team may contain up to three members. Pick a three-letter name for the team based on the first letters of the members' last names. If fewer than three members are on the team, use 'X' for each missing member.

Readme File

You should provide a README.txt text file. Be clear in your instruction on how to build and use the project by providing instructions a novice programmer would understand. If there are any external dependencies for building, the README must also list them and how to find and incorporate them. Usage should include an example invocation. A README would cover the following:

- Class number
- Project number and name
- Team name and members
- Intro (including the algorithm used)
- Contents: Files in the .zip submission
- External Requirements (None?)
- Setup and Installation (if any)
- Sample invocation & results to see
- Features (both included and missing)
- Bugs (if any)

CS 386 — Game Dev — Python Maze

Academic Rules

Correctly and properly attribute all third party material and references, lest points be taken off.

Submission

All Necessary Files: Your submission must, at a minimum, include a plain ASCII text file called **`README.txt`**, all necessary source files to allow the submission to be built and run independently by the instructor. [For this project, no unusual files are expected.] Note, the instructor not use use your IDE or O.S.

Headers: All source code files must include a comment header identifying the author, author's contact info (please, no phone numbers), and a brief description of the file.

No Binaries: Do not include any IDE-specific files, object files, binary **executables**, or other superfluous files.

Project Folder: Place your submission files in a **folder named** `X-pY_teamname`. Where X is the class number and Y is the project number. For example, if your team name is ABC and this is for project #2 in class CS-123, then name the folder `"123-p2_ABC"`.

Project Zip File: Then zip up this folder. Name the .zip file the **same as the folder name**.
Turn in by 11pm on the due date (as specified in the bulletin-board post) by **sending me email** (see the Syllabus for the correct email address) with the zip file attached. The email subject title should include **the folder name**.

ZAP file: If your emailer will not email a .zip file, then change the file extension from .zip to .zap, attach that, and tell me so in the email.

Email Body: Please include your team members' names and campus IDs at the end of the email.

Project Problems: If there is a problem with your project, don't put it in the email body – put it in the README.txt file.

The Cloud: Do not provide a link to Dropbox, Gdrive, or other cloud storage. Note, cloud files (e.g., G-drive) are not accepted.

Grading

- 75% for executing with no errors or warnings
- 10% for clean and well-documented code
- 10% for a clean and reasonable **`README`** file
- 5% for successfully following Submission rules