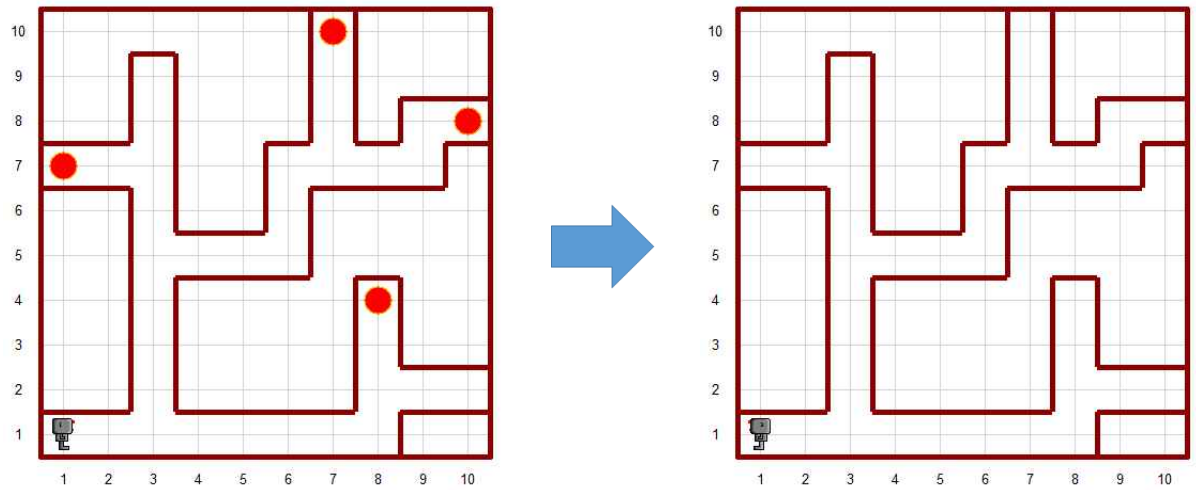
CS101 Homework #1

Defuse Bombs

Due Date: Sunday, March 29, 2015 (Until 23:59)
Delayed Due Date: Wednesday, April 1, 2015 (Until 23:59)

Please read the homework description carefully and make sure that your program meets all the requirements stated. The homework is an individual task. You can discuss the problem with your friends but you must not program together. **You will get F on the entire course if your homework includes any plagiarism.**



Goal

Hubo is actually an FBI agent. He got information that some terrorists planted time bombs in his world. Thus, he moves to defuse the bombs. Let hubo defuse every bomb in the world!

Install *cs1robots2* module

For this homework, we provide the upgraded version of *cs1robots* module.

To make it work, you should do one of the followings.

1) Put *cs1robots2.py* on the directory, which your source code file exists.
2) Put *cs1robots2.py* on *C:/Python27/Lib/site-packages/*

Then, you should import *cs1robots2* instead of *cs1robot*, but don't care if you start with *template.py* file.

In the new module, two functions are added to *Robot* object.

1) **hubo.on_bomb()**: It checks whether hubo is on the bomb or not. The usage is similar to *hubo.on_beeper()*

2) **hubo.defuse_bomb()**: It makes the bomb disappear. Note that it causes error if a robot calls this function when it is not on any bomb. The usage is similar to *hubo.pick_beeper()*

In addition, the bomb can be seen only with the new module. So, please use the provided module.

Requirements

The final goal is to implement a program that makes hubo remove all bombs in the six provided worlds. That is, you should let hubo traverse all ways and call *hubo.defuse_bomb()* on every bomb.

The detail is up to you. However, you can use only what you have learnt until the 3rd lecture. Especially, note that the followings.

1) You CANNOT use function parameters and return values.
2) You CANNOT use *break* or *continue*.

If you have no idea, please read *Hint* section in the below.

Your implementation should work with the provided six maps.

If the maps are too difficult, do your best. Even if your program works on some of the maps, we will give you partial points. You can assume the followings.

1) There is no thick street. Except for the junctions, *hubo.left_is_clear() == False and hubo.right_is_clear() == False* is always satisfied.
2) The bombs are always on the end of the street.

Also, if your implementation uses beepers, please give the maximum number of required beepers in your source code. That is, your program should work with six maps without any changes except for the map file name.

Finally, note that your source code should contain appropriate comments.

Especially, the title of the program and the author information should be written at the head of your source code. In addition, a brief description of step-by-step algorithm should be included as comments at appropriate positions. Moreover, it is highly recommended to add comments to improve the readability of your source code. If your source code contains little comments, you will get some penalties.

You also should submit a report.

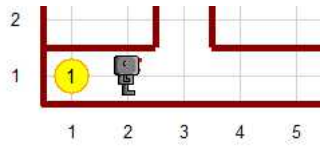
There is no strict format, but the report should include the following things. Please note that only *doc*, *docx*, and *pdf* files are allowed for your report. Do not submit any *hwp* file.

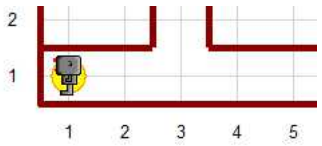
1) Summary or explanation about the overall algorithm.
2) Screen shots of your program
- Choose screen shots which can show processes and results of your program execution, well.
- (e.g.) Initial step, intermediate steps, end step, and etc... (2~4 images are enough.)
3) Describe about what you learn and what you feel while doing homework.

The report may contain parts of your source code, but please do not include whole source code.

Hints

The most difficult part may be specifying terminal condition of the program. Thus, we provide the related code from a solution as a template code.





At the initial position, hubo drops a beeper and moves on. (the left figure). After defusing every bomb, hubo comes back to the initial position. Then, hubo will be on a beeper, but front, left, and right is not clear. (the right figure) This is the terminal condition that is typed in the template code.

In addition, we provide two video files. The videos show the behavior of a sample solution we made. The main strategy of the solution is to drop beepers on junctions we are exploring.

If hubo meets a new junction, it drops three beepers. This means that three directions (right, front, left) have not yet been explored. Then, go to the right direction. After exploring the right direction, hubo comes back to the junction again, and it picks a beeper. Now, there are two beepers left on the junction. This means that only two directions (front, left) have not yet been explored. After picking the last beeper, hubo can conclude to complete the junction, and go back.

In other words, when hubo reaches a junction, there are four cases.

1) No beeper: Hubo has not yet been on the junction. That is, a new junction.
2) 3 beepers: Hubo finishes exploring the right direction. Try to explore the front direction.
3) 2 beepers: Hubo finishes exploring the right and front direction.
4) 1 beepers: Hubo just finishes exploring the right, front, and left direction. Go back.

Submission

You need to submit the followings:

- The file "HW1_studentid.py"
(e.g.) HW1_20150215.py

- The report "HW1_yourid.doc" or "HW1_yourid.docx" or "HW1_yourid.pdf"
(e.g.) HW1_20150215.doc, HW1_20150215.docx, HW1_20150215.pdf

You MUST archive the source code and the report together into "HW1_yourid.zip" and submit the archived file through the webpage for homework submission.
(e.g.) HW1_20150215.zip

If you don't follow the submission policy, you will get penalty.