CS101 Homework #1

Hubonity

Deadline: September 30th, 2024 (23:59 KST)

Please read the homework description carefully and ensure your program meets all the requirements stated. Since homework is an individual task, you **MUST** write your own code. **If you plagiarize someone's work, even if it's a single line or word, both you and the provider of the code will get an F for the entire course. You will also get F if you use the code generated by AI chatbot service**.

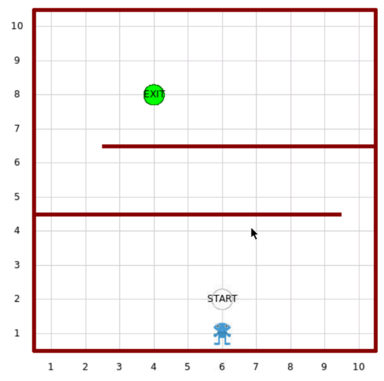
- **DO NOT** write code outside the allowed areas: `STUDENT_ID_NUMBER` and `robot_control()`. Otherwise, your program may not be properly graded, and the result will be irreversible.
- Your code **MUST** be written and submitted to Elice. You cannot earn any points by simply running your code. The grader grades your code only after you click the submit button; grading will be done automatically. You can resubmit as many times as you wish. Your homework score will be the score of your latest submissions. **No late submission is allowed**.
- You **MUST** use "Forums - HW1 Q&A" in Elice to ask questions or put comments while you **DO NOT** reveal your code. TAs can always access your code via Elice. **TAs WILL NOT answer any contact via other channels**.
- **DO NOT** use any additional Python external library except `cs1robots`.

Homework 1 consists of five tasks (Task 1-1 to 1-5) that need to be implemented and graded separately. Each task is worth 5 points, so you can obtain a maximum of 25 points.

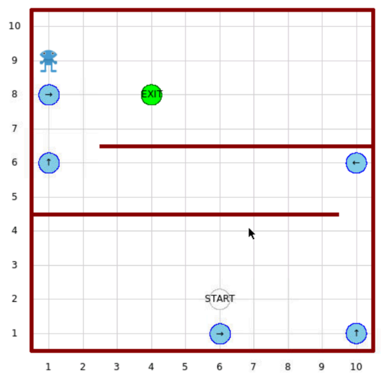
Overview

In this homework, you **MUST** reassign the global integer variable `STUDENT_ID_NUMBER` with your 8-digit student ID. Then, `generate_world_and_robot` takes `STUDENT_ID_NUMBER` as a seed of the random module and generates the world and `hubo` in the `main` function. It means that `generate_world_and_robot` generates the same world if the value of `STUDENT_ID_NUMBER` is the same.

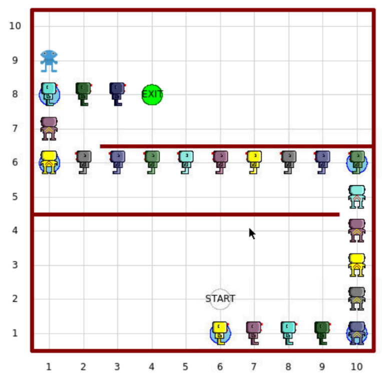
The theme of this homework is an escape puzzle. In the special version of `cs1robots` library provided in Homework 1, `hubo` is a special `Robot` that can control other robots called *minions*. After your `robot_control` function is finished, `generate_minions` alternates generating a minion at *start* and making every minion move forward. You should write the `robot_control` function that controls `hubo` to place a set of *commands* to guide your minions to the *exit*. If your minions reach the exit, you get the points. However, your points can be forfeited if you modify a given map, change `hubo`'s location without using `hubo`'s `move` function, or assign a student ID number that does not belong to you.



1. initial state



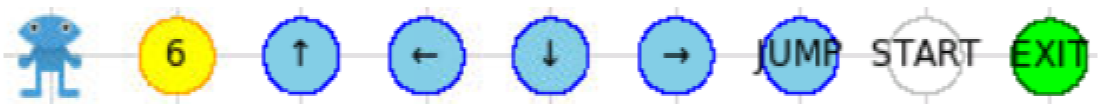
2. place the commands



3. minions follow them

Description

In this homework, `cs1robots` is a special version that contains commands, a start, and an exit. The following figures show hubo, a beeper, commands, a start and an exit.



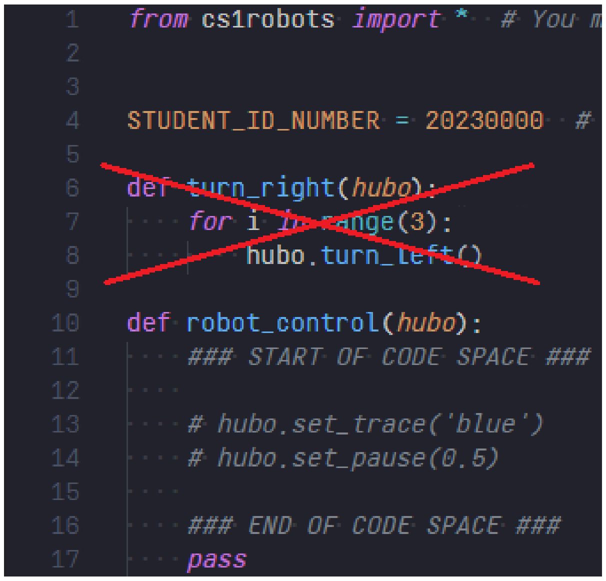
hubo / beeper / **north (1)** / **west (2)** / **south (3)** / **east (4)** / **jump (5)** / **start** / **exit**

- **hubo** - You can decide its behavior by writing the `robot_control` function. It is almost the same as normal Robot objects, but the beepers dropped by `hubo` become a **command** by their number. The initial position and orientation are random.
- **command** - A direction command forces minions' direction as its direction. After a `Robot` object's `move`, if it is on a jump command, it moves again, ignoring walls. In other words, the jump command affects both `hubo` and minions.
- **beeper** - If the number of beepers does not match with any command, they are just beepers.
- **start** - Where the `generate_minions` function generates minions. Its position and initial direction is provided by the `generate_world_and_robot` function, and you cannot modify it.
- **exit** - An exit indicates the destination of each task. If minions have arrived at the exit, you get the score. Its position is provided by the `generate_world_and_robot` function, and you cannot modify it.

The special `cs1robots` provides `generate_world_and_robot()` function, which generates robot worlds and `hubo`, and returns `hubo` and start config. You **MUST** assign your 8-digit student ID number as an integer to `STUDENT_ID_NUMBER` so that the robot world that belongs to you can be generated.

Note

- Since you can modify only `STUDENT_ID_NUMBER` and `robot_control()`, you **MUST NOT** define your own function in the global space. Even if you do, that function cannot access `hubo` because `hubo` is a local variable. Therefore, you need to define your function locally. The following figure is an example of a locally defined function.

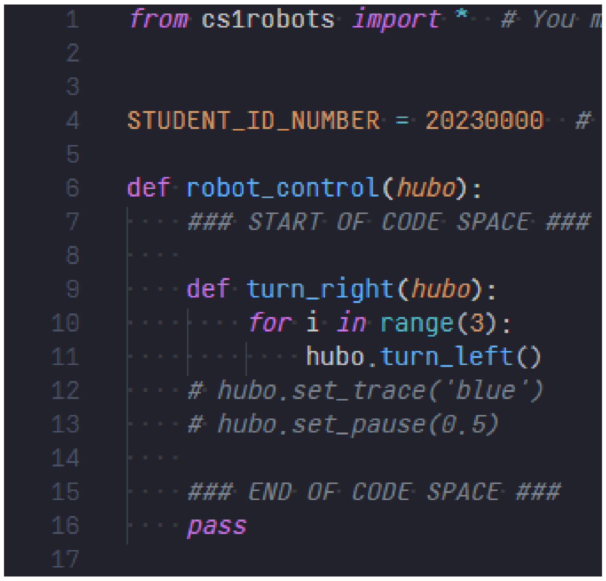


```
from cs1robots import *  # You m


STUDENT_ID_NUMBER = 20230000  #

~~def turn_right(hubo):~~
~~    for i in range(3):~~
~~        hubo.turn_left()~~

def robot_control(hubo):
    ### START OF CODE SPACE ###
    
    # hubo.set_trace('blue')
    # hubo.set_pause(0.5)
    
    ### END OF CODE SPACE ###
    pass
```



```
from cs1robots import *  # You m


STUDENT_ID_NUMBER = 20230000  #

def robot_control(hubo):
    ### START OF CODE SPACE ###
    
    def turn_right(hubo):
        for i in range(3):
            hubo.turn_left()
    # hubo.set_trace('blue')
    # hubo.set_pause(0.5)
    
    ### END OF CODE SPACE ###
    pass

```

- Even if the grader prints a pass for your code, it takes some time to send your score to the server. Therefore, we recommend you to wait until your submission is finalized.
- After submitting your code, Elice may display an animation of something that holds an X sign. It does not mean you have failed on the task. Elice displays it whenever the points you obtain are not 100.