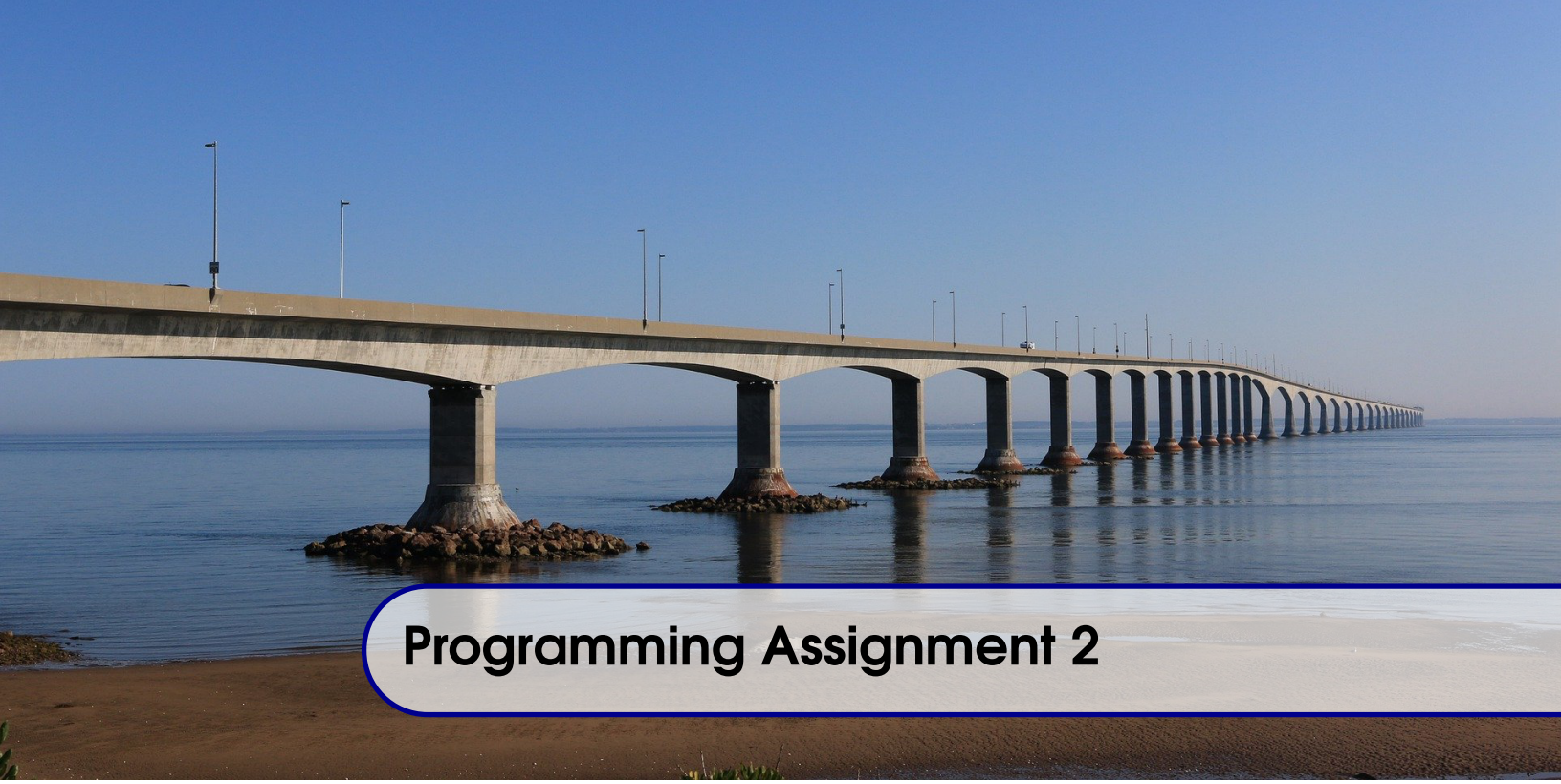

Programming Assignment 2

Assignment Objectives

By completing this assignment you are demonstrating your ability to:

- Decompose problems to create new solutions
- Call functions
- Apply logical reasoning through conditionals and Boolean expressions
- Apply repetitions to iterate over a pattern of computations

Scenario

In this course you will have 4 assignments, all of which are going to be related to the same topic: Maze!

Maze Introduction

A $n \times n$ maze is a collection of paths among interconnected cells in a grid, where n is the number of rows and/or the number of columns. The classic maze path-finding problem tries to find a feasible path from a source cell to a destination cell. Any cell could be of one of the three possible types: (i) accessible cell: a cell which allows a path to pass through it, (ii) blocked cell: a cell that acts as a dead-end and blocks a path or, (iii) null cell: a cell that is neither accessible nor blocked; a null cell is used to initialize and create the maze. In this assignment, we just need to worry about the accessible cells and the blocked cells. Any valid path from a source to a destination can only pass through the accessible cells. In this assignment, we have a better maze-generator that can produce a random maze of size $n \times n$, where $n >= 3$. As shown in figure 1, the first cell has coordinates $(0,0)$. Note that in Python and most of the other programming languages, indexing starts from 0 (and not 1).

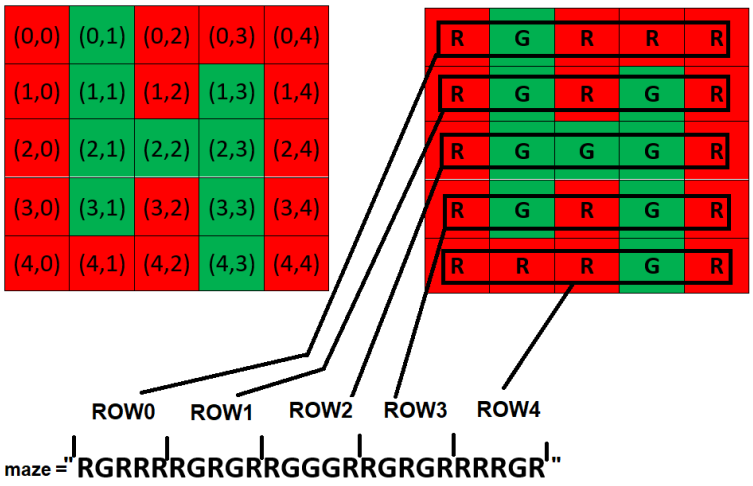

Figure 1: A random 5×5 maze

The "maze" module

To help you solve the problems from the assignments we have created a *maze* module with diverse functionalities that you can use. For using the module, download the file *maze.py* and import it into your program.

How to import the "*maze*" module and call its functions:
As an example assume that you are writing a program in *main.py* under your PyCharm project
Download *maze.py* and put it inside your PyCharm project (on the same path where *main.py* is)
Write the following as the first line of your program (in *main.py*)

import maze as mz

call the function function_name of the "maze" module

mz.function_name(arguments)

Note: in your assignment you will have a .py file for each problem replacing *main.py* in the example.

What functions are there in the "maze" module:

print_maze(maze)

Description: This function displays the maze. The accessible cells are labelled "G" i.e. green, the blocked cells are labelled "R" i.e. red, and the null cells are labelled "Y" i.e. yellow. The argument "maze" is a string over an alphabet set "G", "R", "Y" that encodes the cell labels of the maze. Row labels of the maze are concatenated to form a single string that represents the maze, see figure 1.

create_maze(n)

Description: This function creates and returns a "maze" string of size $n \times n$. The value of an argument n should be $>= 3$ for this function to work. This function returns a string "maze".

get_source()

Description: This function returns the coordinates of the source cell in the format "*source_x:a,source_y:b*". Here *a, b* are the *x, y* coordinates of the source cell.

Example:

source_x:0,source_y:1 represents a source cell at coordinates $(0,1)$.

get_destination()
Description: This function returns the coordinates of the source cell in the format "*destin_x:a,destin_y:b*". Here *a, b* are the *x, y* coordinates of the destination cell.
Example:
destin_x:4,destin_y:3 represents a destination cell at coordinates $(4,3)$.

Problem 1

Write a program in Python that performs the following tasks in order:

1. Takes the size of maze as an input from the user and assign a variable name "*n*" to this value.
2. Our "maze" module can create a random maze of size $n \times n$, where $n >= 3$. It will result in an error if $n <= 2$. In this assignment, we will also impose an upper limit on the maze size i.e. $n > 9$ will make the maze invalid. In your program, validate the value of *n* i.e.
   (a) Print "Invalid value" if the value of *n* is greater than 9 or the value of *n* is less than or equals to 0.
   (b) If the value of n is 1, then print "Source and destination cells are the same".
   (c) If the value of n is 2, then print "Destination is reachable from the source".
   (d) If the value of n is greater than or equals to 3 but less than 10, then:
      i. Create a maze using the function(s) of the given "maze" module.
      ii. Print the maze produced in the previous step using function(s) of the given "maze" module.
      iii. Compute and print the number of green cells and the number of red cells in your maze.

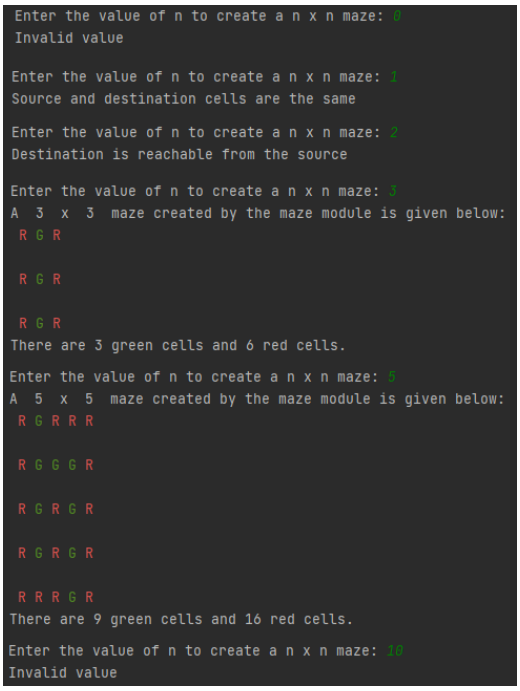

Figure 2: Sample program runs of completed Problem 1.

Problem 2

This problem is the continuation of problem 1. You have already created and printed a maze in problem 1. Using the maze produced by the functions of the "maze" module, write a program in Python that performs the following tasks in order:

1. Get the coordinates of the source cell using the function(s) of the given "maze" module and print these coordinates in the format as shown in the fig 3. Because the function from the "maze" module will return the coordinates of the source cell as a string, you need to use repetitions to extract these coordinates from the returned string before you print them in the required format.
2. Get the coordinates of the destination cell using the function(s) of the given "maze" module and print these coordinates in the format as shown in the fig 3. Because the function from the "maze" module will return the coordinates of the destination cell as a string, you need to use repetitions to extract these coordinates from the returned string before you print them in the required format.
3. Compute and print the following using repetitions (iterating through the elements of the maze string):
   (a) number of green cells in the row that contains the source cell
   (b) number of red cells in the column that contains the destination cell

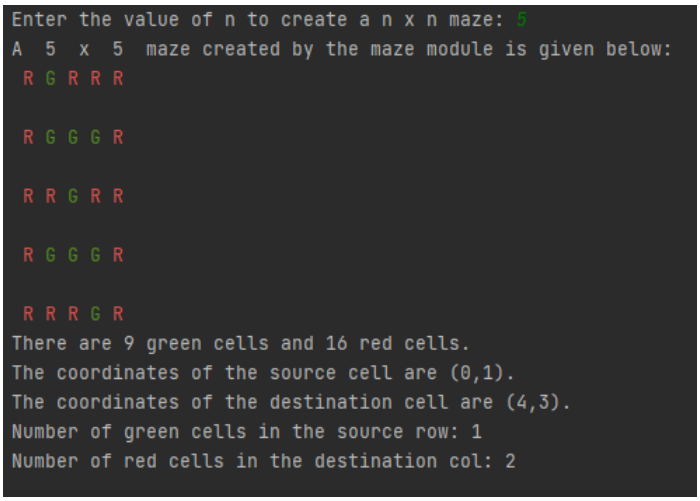

Figure 3: Sample program run of completed Problem 2.

Submitting your assignment

Your submission will consist of one zip file containing solutions to all problems (see below). This file will be submitted via a link provided on our Moodle page just below the assignment description. This file must be uploaded by 5pm (Charlottetown time) on the due date in order to be accepted. You can submit your solution any number of times prior to the cutoff time (each upload overwrites the previous one). Therefore, you can (and should) practice uploading your solution and verifying that it has arrived correctly.

What's in your zip file?

A zip file is a compressed file that can contain any number of folders and files. Name your zip file using this format.

Example:
Submission file: `asnX_studentnum.zip`
Where `X` is the assignment number between 1 and 4, and `studentnum` is your student ID number.

Your zip file should contain a PyCharm project. The project folder should contain all Python files for the successful execution of your programs. Each problem should have a file named `problem#.py` where the # is the problem number.