

# COSC 101 Homework 4: Fall 2024

The due date for this homework is **Thursday, October 10th, 11:59 pm EDT**

## Introduction

This assignment is designed to give you practice with the following topics:

- Procedural decomposition and composition
- **for** loops with **range** function
- Using a custom module

## Your assignment

In the first lab, you practiced computational thinking by writing instructions to build a simple model out of Duplos. In homework 2, you practiced your procedural decomposition and composition skills by writing a program to draw a house as text art. Now, you'll use your burgeoning programming skills to **design and partially implement a program to automatically generate building instructions for a house built out of Legos**.

Unlike prior assignments, where each task was separate, the **tasks in this assignment build on one another**. You need to complete Task 1 before you can do Task 2, and you need to complete Task 2 before you can do Task 3.

## Task 1: Problem Decomposition and Pattern Recognition

Writing a program that solves a (somewhat) complex problem requires thinking carefully about the problem and generating a rough plan for how you will structure your program. You may slightly deviate from your plan once you start programming, because you'll learn more about what works (or doesn't work) as you write actual code. However, if you just dive right in and start programming without any plan, you'll likely spend a lot more time writing (and re-writing) code that doesn't work.

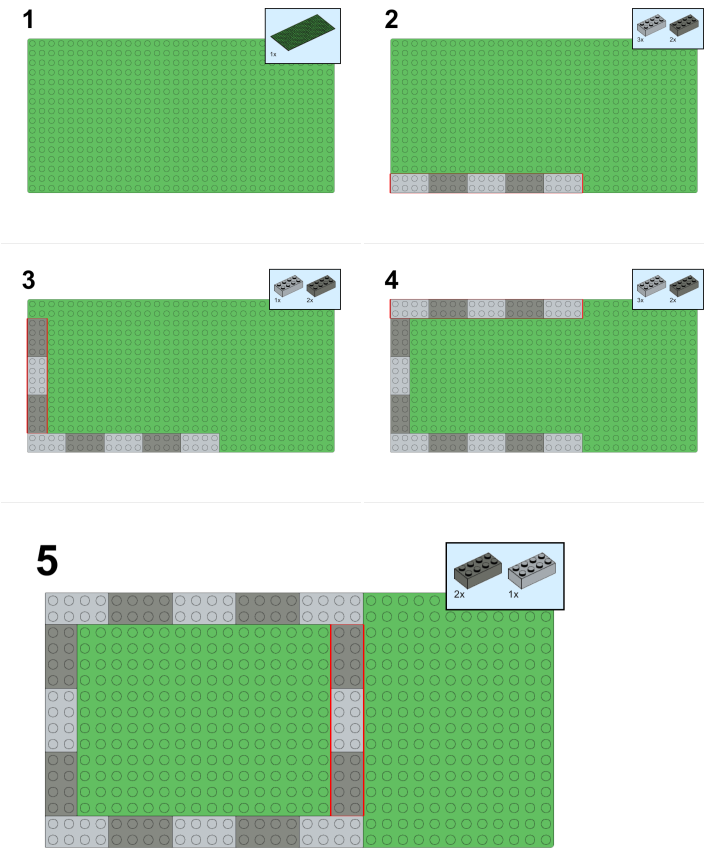
The basic unit of measurement in Legos is a stud. A standard Lego brick is 2 studs wide by 4 studs long. To keep things simple, we'll limit our focus to houses whose length and width (in studs) are **multiples of 4**, which can be easily constructed using standard 2x4 bricks (and a few 2x2 ones if openings like door and windows are included). For example, we will go through the process of building a house that is 20 studs long and 16 studs wide.

### Foundation

The first step in building a house is to build a foundation. The foundation defines the overall size of the house. A Lego baseplate is like a plot of land or grid on which we can build the foundation. Lego half baseplates are 16 studs x 32 studs.

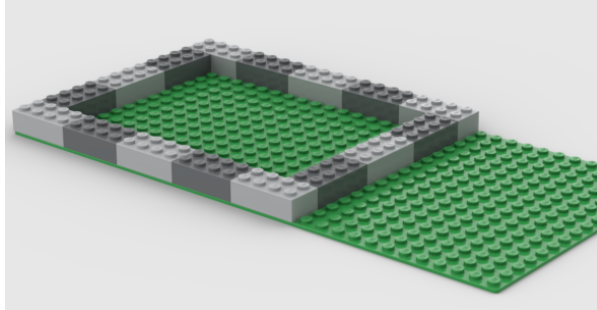
The foundation—like the entire house—is built one brick at a time. However, Lego instructions typically combine the laying of multiple bricks into a single step. Combining too many bricks into one step is confusing, so we can think of logical groupings of bricks as a single step. For example, the steps below

show a top view of the house with the laying of the foundation one side of the house at a time (front, left, back, right).



If we think of the baseplate like an x-y grid, where the stud in the lower-left corner is (0, 0), then we can think of placing each brick at certain coordinates in the grid. For example, the lower-left corner of the leftmost light gray brick in step 2 is located at (0, 0), while the lower-left corner of the leftmost dark gray brick in step 2 is located at (4, 0).

Here is a 3D view of our current house.



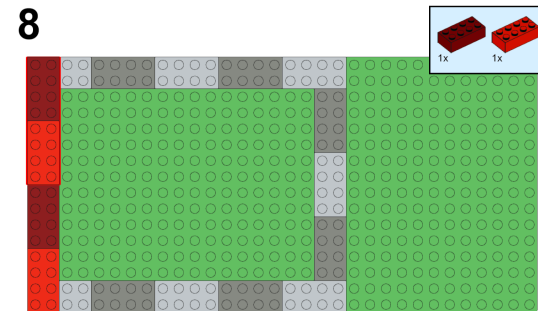
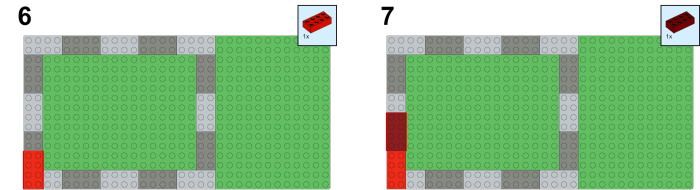
**Subtask A: Answer the following questions:** *You can write your answers on paper and scan them or create a Google doc. Regardless of which method you choose, you will submit a single PDF file with your answers to all of the questions in Task 1.*

1. Label each brick in steps 2 through 4 with the coordinate of the lower-left corner of the brick. *If you complete your work on paper, you will need to print out the steps to annotate them. If you complete your work in a Google doc, you will need to copy the images into your Google Doc.*
2. What pattern do you observe among the coordinates for the bricks in the front foundation (added in step 2)?
3. What pattern do you observe among the coordinates for the bricks in the back foundation (added in step 4)? How does this compare to the pattern you observed in the front foundation?
4. What pattern do observe among the coordinates for the bricks in the left side foundation (added in step 3) and the right side foundation (added in step 5)?

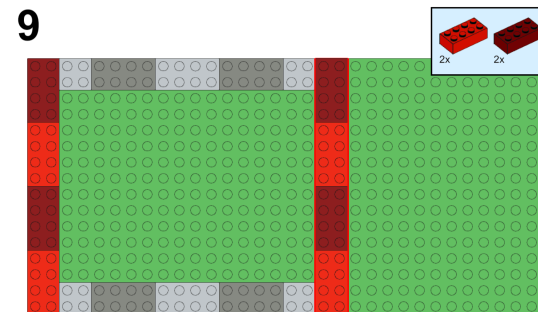
#### First course

Now that the foundation is laid it's time to build the first **course** of the house. (We'll call the foundation the zeroth course.)

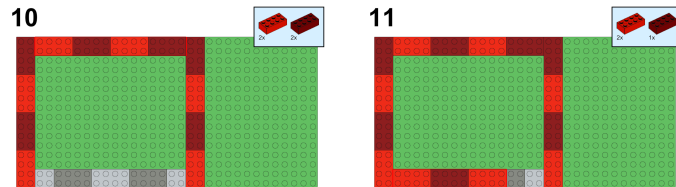
To build a strong house, the bricks should follow a *running bond* pattern, in which each brick in the first course is offset by a half a brick from the previous (zeroth) course. For example, notice how the red brick added in step 6 overlaps half of the light gray brick in the front foundation and half of the dark gray brick in the side foundation. Similarly notice how the dark red brick added in step 7 overlaps the remaining half of the dark gray brick in the side foundation and half of the adjacent light gray brick in the side foundation. Step 8 adds the remaining bricks for the first course of the left side of the house.



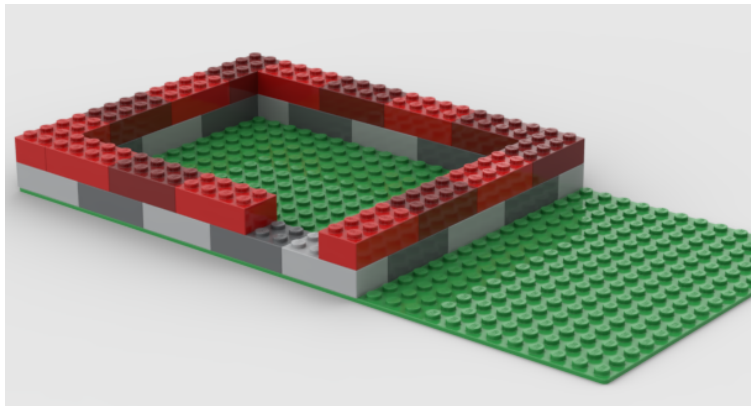
The bricks in the first course on the right side of the house following the same running bond pattern.



The bricks in the first course on the back and front of the house also follow a running bond pattern. However, one brick is omitted in the front to leave a doorway.



Here is a 3D view of our current house.

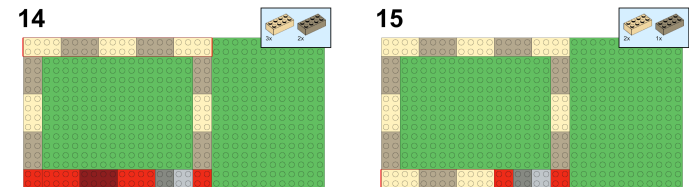
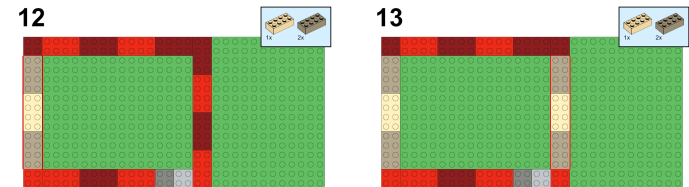


#### Subtask B: Answer the following questions:

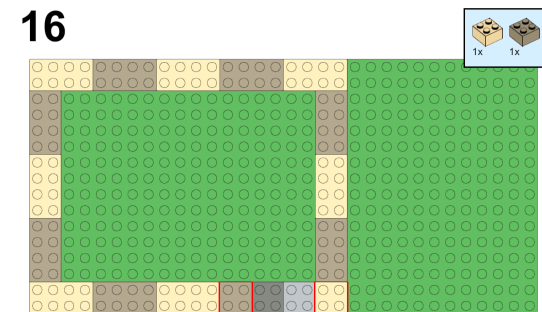
- Label each brick in steps 6 through 11 with the coordinate of the lower-left corner of the brick.  
*If you complete your work on paper, you will need to print out the steps to annotate them. If you complete your work in a Google doc, you will need to copy the images into your Google Doc.*
- Recall that bricks must be placed one at a time, even though they are grouped into steps. How is the process for placing bricks one at a time different for the first course on the back of the house compared to the first course on the front of the house?
- If you used a `for` loop to "solve" the problem of placing bricks, how would the `for` loop for the first course on the back of the house be different from the `for` loop for the first course on the front of the house?

#### Second course

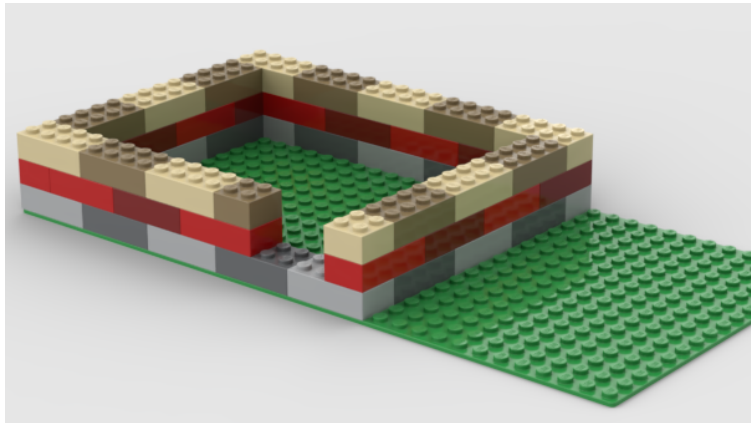
The second course continues the running bond pattern. However, we need to adapt the pattern around the door, otherwise our bricks will block the doorway.



In particular, we need to use two 2x2 bricks, instead of two 2x4 bricks, to finish the second course of the front of the house.



Here is a 3D view of our current house with the foundation, first course, and second course.

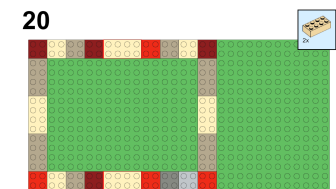
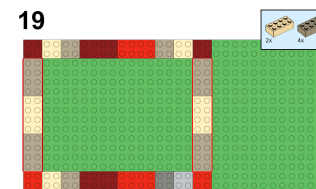
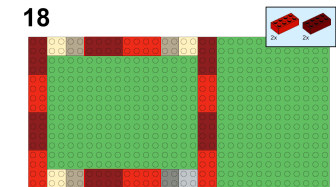
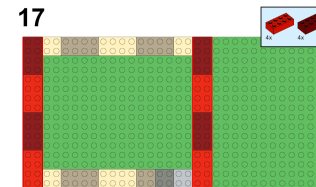


#### Subtask C: Answer the following questions:

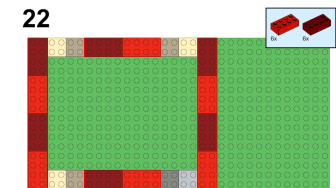
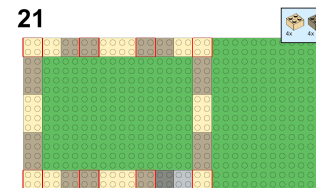
8. If you used a `for` loop to "solve" the problem of placing **2x4 bricks**, how would the `for` loop for the second course on the back of the house be different from the `for` loop for the second course on the front of the house?
9. Look back through the steps that have occurred so far. If you had to divide the steps into three functions, how would you divide them?
10. Look back through the steps that have occurred so far. Which steps are very similar?

#### Courses with Windows

Let's add one more detail to our house: windows. In particular, we'll leave some gaps in the third, fourth, and fifth courses for windows, similar to how we left a gap for the doorway in the first and second course. Since there is already a doorway in the front right corner of the house, we'll put windows in the front left corner, back left corner, and back right corner. We'll also remember to leave the doorway clear, since no minifig (the name for Lego people) wants a doorway they have to crawl through!

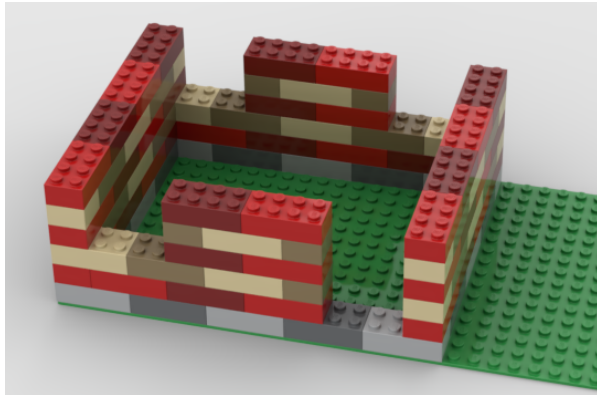


In the fourth course, we'll again need to use some 2x2 bricks, so we don't block our doorway or windows.



Here is a 3D view of our current house with the foundation, first and second course, and courses with windows.





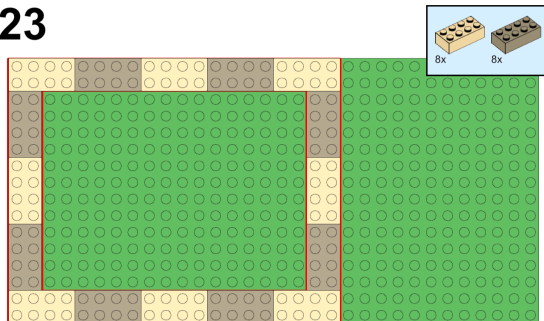
#### Subtask D: Answer the following questions:

11. Look back at your answer to question 9 in subtask C. If you were to add three more functions, what steps would each function encompass?
12. Which previous steps (between steps 1 through 16) are most similar to steps 17 and 19?
13. Which previous steps (between steps 1 through 16) are most similar to steps 18 and 20?

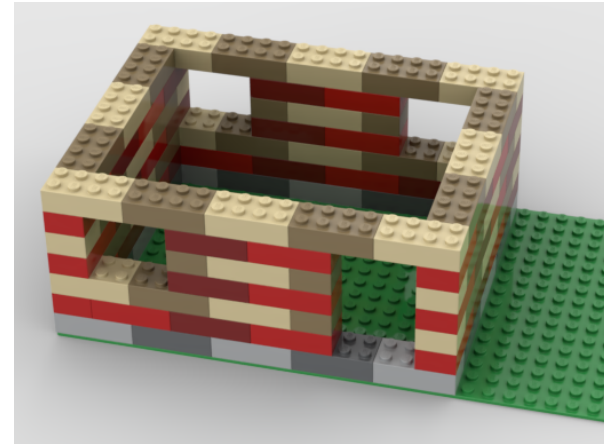
#### Final Course

We'll end the house with one more course of bricks.

# 23



Here is a 3D view of our final house. (A roof would be a good addition, but there's already enough complexity that we won't worry about this.)



#### Subtask E: Answer the following questions:

14. Which previous step(s) are most similar to step 23?

### Task 2: Prototyping

At this point, you should have a complete understanding of the problem we are trying to solve: generating instructions for building a house out of Legos. Now, you'll work on prototyping part of the solution. In particular, we'll focus on writing code to create the building instructions for each course of the front wall, because the front wall of the house contains almost all of the patterns we need to handle.

#### Using the custom `instructions` module

To assist you in drawing the building instructions, we have written a custom `instructions` module. The module allows you to:

- Create a new course
- Place bricks on a course
- Display a course with the bricks that have currently been placed

Real world programmers often need to use/interface with code they did not write. Hence, being able to use some existing code that you have not used before is an important skill for you to develop.

To use the custom module, you must add `import instructions` to the `hw4_prototype.py` file.

Calling the module's `create_step` function will return a new `Step` object you can use in your code. Objects are custom collections of values (e.g., integers, strings, lists) and related functions. A `Step` object tracks the size of the build area, which bricks are part of the current step, and where they have been placed.

Calling the `create_step` function is similar to calling functions in other modules like `math` and `random`. The `create_step` function takes two parameters: the length of the build area and the width of the build area. Here is an example:

```
step = instructions.create_step(8, 6)
```

You can use a `Step` object's `place_brick` function to add a brick to the step. The function takes four parameters:

- The x-coordinate where the lower-left corner of the brick should be placed
- The y-coordinate where the lower-left corner of the brick should be placed
- The length of the brick to place
- The width of the brick to place

Here is an example that places a 2x4 brick in the lower-left corner of the build area:

```
step.place_brick(0, 0, 2, 4)
```

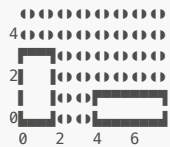
Here is another example that places a 2x4 brick in a different orientation in the lower-right corner of the build area:

```
step.place_brick(4, 0, 4, 2)
```

You can use a **Step** object's **display** function to display the step, which will produce a text-based drawing that is similar to the images in Task 1. For example after the 3 calls above the call below:

```
step.display()
```

will produce the following output



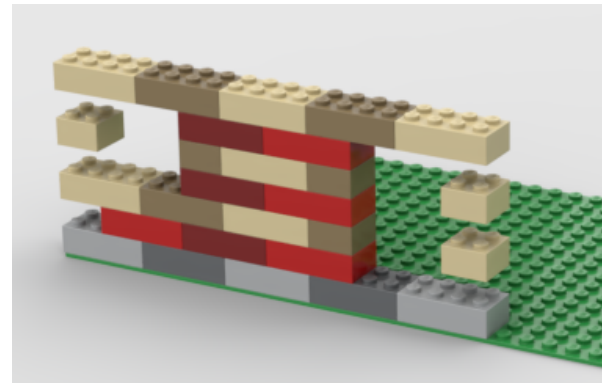
### Subtask A:

1. Add code to the `front_foundation` method in `hw4_prototype.py` to reproduce the step 2 drawing in Task 1. Your code should contain multiple calls to `place_bricks`; don't use a `for`

- loop. You may want to refer to your answer to question 1 in Task 1 to assist you.
- Comment out (do **not** delete) the calls to `place_bricks` you added to the `front_foundation` method. Replace these statements with a `for` loop that calls `place_bricks`. You may want to refer to your answer to question 2 in Task 1 to assist you.

### Instructions for Front Wall Patterns

**Subtask B:** . Add code to the remaining `front` functions in `hw4_prototype.py` to create an instruction step for each course of the front wall. Bricks that are fully contained within the bottom two studs of the building area are considered part of the front wall. Here is a rendering of a complete front wall:



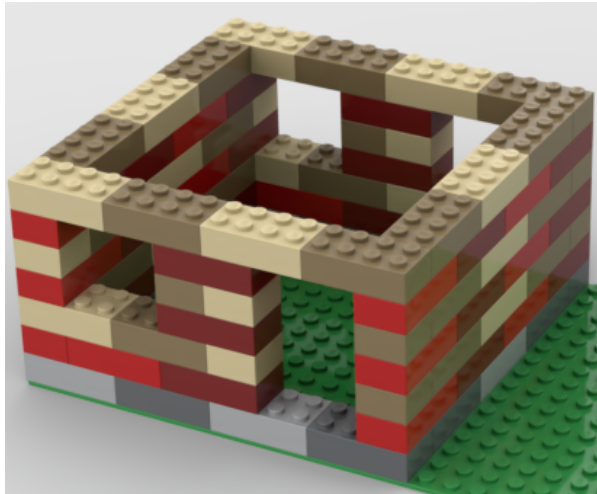
You may want to refer to the drawings and your answers to the questions in Task 1 to assist you. You may **not** modify the code in `main`. Also, you should **avoid duplicating the exact same code in two functions**, so you may need to add some additional "helper" functions that are called from two or more of the existing `front` functions.

### Task 3: Generalizing

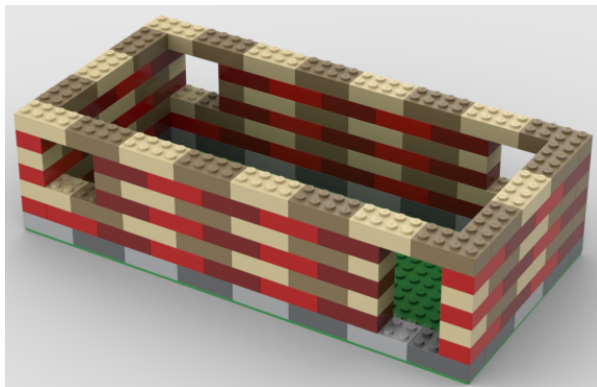
Now that you have written functions that work for the front wall of a fixed size house, it is time to generalize your code to handle different variations of the problem.

In particular, you will generalize your code to work with houses of variable lengths. Task 1 stepped through the process of building a house that is 20 studs long, but the process can easily be generalized to any length that is a multiple of 4 and at least 16 studs.

For example, here is a house that is 16 studs long:



As another example, here is a house that is 32 studs long:



#### Final Task:

1. Copy all of your code from `hw4_prototype.py` into `hw4_generalized.py`
2. Modify the `main` function (in `hw4_generalized.py`) to ask the user "How long is the house?" If the user enters a value that is less than 16, the program should output "Too short" and terminate. If the user enters a value that is not a multiple of 4, the program should

output "Must be a multiple of 4" and terminate. Otherwise, the program should display the building instructions for each course of the front wall.

3. Update your `front` functions (in `hw4_generalized.py`) to display the proper instructions for each step based on the length the user entered. When you call `create_step`, you should use the length entered by the user as the first parameter, instead of a fixed size of 32.

#### Submission Instructions

Submit a single PDF with your answers for Task 1 and the Python files from Task 2 (`hw4_prototype.py`) and Task 3 (`hw4_generalized.py`) to the platform indicated in your class section.

Remember to complete the questions at the top of each Python file before submitting.