The objectives of this exercise:

- **Computational:**
  - Algorithmic thinking: (1) As a designer, developing a set of step by step instructions to generate a visual pattern on a base grid from straight line segments, and (2) As a responder to another group’s instructions, carry out their instructions methodically
  - Abstraction: Using relative rather than absolute coordinates to generate the pattern instructions
  - Decomposition: Breaking down a 9x9 grid into simpler 3x3 components
  - Pattern recognition: Pattern Recognition: As a designer, identifying similar (perhaps repeated) steps that create similar visual patterns on the grid and then using loops to simplify those steps, and (2) As a responder to another group’s instructions, identifying similar (perhaps repeated) steps that create similar visual patterns on the grid, and using short cuts to carry out those steps faster
  - Evaluation: Testing your instructions to make sure that another user can accurately recreate your pattern on a new base grid
  - Learning about the benefits of looping by using looping to simplify the instructions and allow patterns to be repeated
  - Learning how to design modular functions by modifying the instructions to use *relative* coordinates
  - Further developing testing skills by using the modular function to generate a more complex pattern

- **Creative:**
  - Surrounding: using your senses of touch and sight to follow a set of instructions and perceive that they produce the intended geometric pattern
  - Capturing: creating new outputs and using new ways to represent and save data by writing a description of a path which will generate a drawing of a geometric pattern
  - Challenging: looking at written descriptions in new ways as you both generate and follow them to recreate a drawing of a geometric design
Broadening: acquiring new information and skills by understanding how simple rules can generate complex patterns

Collaborative:
- Being open to all points of view and resolving conflicts in a constructive way.
- Giving and receiving thoughtful and constructive feedback in order to develop your group project.
- Meeting group deadlines, including completing your individual work in a timely manner.
- Contributing substantively to the group process, using your skills, knowledge and experience.
- Working together as a team to achieve a common goal; being able to both compete against and cooperate with other teams.

**Problem Description**

Note: This exercise is an extension of the Exercise Path Finding I. Please refer to that for the problem description.

Each group will set up a wiki page on agora.unl.edu. The name of this page should be: “Path Finding II by <Course> Group <Name>” where <Course> is the course abbreviation and <Name> is your group name (e.g., Path Finding II by CSCE 155A Group Awesome).

**Any member may create the group page. Note that there should be only one page created per group. Before you create a new page, make sure that one doesn’t already exist.**

1. **Week One [20 points]**
   Using a base 3 x 3 pattern from Exercise Path Finding I, build a complex 9 x 9 pattern using three operations: (1) shift, (2) reflect, and (3) rotate. The *shift* operation allows a base pattern to be replicated at a different location. The *reflect* operation allows a base pattern to be reflected horizontally or vertically. The *rotate* operation allows a base pattern to be rotated 90 degrees clockwise or counterclockwise. Figure 1 shows examples of these operations performed on a base pattern.
The use of looping to allow for repetition in functions is extremely important in computer science. Loop structures allow similar operations (or instructions) to be repeated inside a function using a compact representation (in the source code). Obviously, this is very convenient for programmers who do not want to cut and paste the same source code multiple times. This compact representation also reduces the size of the final program, which can be useful when hardware resources are limited, for example, on an interplanetary probe or nano machine. However, the real benefits of using loop structures come into play when these similar operations need to be modified. Suppose the application you are developing is a Facebook game with 999 different ways to configure character appearance. With a looping structure, you may only need to change the rendering engine in a single place, potentially saving you 998 extra changes. Of course, designing the operations to be similar enough to take advantage of looping structures, while still satisfying the design features is not always easy. Creative thinking skills during the design process can help the programmer leverage looping structures.

![Figure 1: Examples of Base Pattern along with Reflect, Rotate and Shift Down.](image)

Note that depending on your pattern, rotating or reflecting may produce the same visual result. You are also allowed to use a loop structure to repeat to generate multiple shifts, reflections, or rotations. For example,

```plaintext
Loop 3 times

    Rotate-counter-clockwise()

End Loop
```

In the above example, a base pattern will be rotated 3 times counter-clockwise.

💡
**IMPORTANT:** Note that in order to do shifting, reflection, or rotation operations, the base pattern’s instructions cannot use **absolute references** for the coordinates. Instead, **relative coordinates** are needed.

To illustrate the need for relative coordinates, suppose we want to move three segments horizontally. The instruction to move from 0,0 to 3,0 (with the first coordinate as horizontal and the second as vertical) works fine for base pattern, but will not work after we shift and start drawing at 4,0 because it uses absolute references (0,0 and 3,0). Instead, we need to modify the instruction to go from the current coordinates \(x,y\) to \(x+3,y\). This instruction uses relative coordinates \((x,y \to x+3,y)\). Using these relative coordinates will give the desired end coordinates for the base pattern when \(x=0, y=0\) and also for the shifted pattern when \(x=4, y=0\). Now, suppose that in addition to shifting, we want to rotate the base pattern 90 degrees. After the rotation, we need to move vertically rather than horizontally. Again, modifying the instruction requires only a small change when using relative coordinates: \((x,y \to x,y+3)\).

Thus, before you start performing the above operations on the base pattern, please revise the instruction set for the base pattern to use only relative coordinates. *(Hint: Think of using variables!)*

**IMPORTANT:** In Exercise Path Finding I, the number of line segments in the base pattern had to be at least 30% of all possible lines in a 3 x 3 grid. **Now, the number of line segments in your complex pattern must be at least 30% of all possible lines in a 9 x 9 grid.** Note that a 9 x 9 grid can have at most 90 horizontal segments, 90 vertical segments, and 162 diagonal segments for a total of 342 line segments. After rounding up, this means that your complex pattern must have at least 103 segments.

**NOTE:** Teams submitting patterns with less than 103 segments will lose points.

**NOTE:** Teams without rotating, reflecting AND shifting elements will lose points.

**IMPORTANT:** Your written instructions for generating the complex 9 x 9 pattern must be sufficiently clear to enable another group to accurately draw your pattern on a similar grid. You will lose points if the grader or another group cannot follow your instructions.
The use of relative coordinates, and, in extension, variables, is important for practicing computer science. Functions using relative coordinates provide several benefits compared to using those absolute coordinates. First, a function using relative coordinates is more flexible than a function using absolute coordinates. The following examples show how relative coordinate flexibility is useful for both the user and the machine: (1) the user can run a function when he/she is at a similar, but not exactly the same, starting point (e.g., at 0,1 rather than 0,0) and (2) the machine can assign the function to available addresses in the memory rather than waiting until absolute addresses are available. Second, a function using relative coordinates is much easier to modify than one using absolute coordinates. In functions using absolute coordinates, since the exact coordinates are passed from one line to the next, a change to the coordinates on one line must be propagated throughout the rest of the function. This can be extremely time-consuming and error-prone on functions with hundreds (or thousands) of lines. On the other hand, such a change to functions using relative coordinates generally only affects a single line—making them easier to modify. The benefits of using relative rather than absolute coordinates also extend to using variables over hard-coded values. In general, functions using variables are more flexible and easier to modify than hard-coded values. However, these functions can also be more difficult to design since they require a programmer who can think creatively about how to write each line using variables rather than hard-coded values.

Each group should navigate to a different group’s wiki page and “claim” their pattern instructions by posting on their wiki page. (Each group’s pattern instructions may have only ONE group attempting to follow it.) Each group will then try to follow the written instructions of the other group by drawing on a blank grid labeled following the same (0,0) numbering. When you think you have followed the other group’s instructions correctly, post an image of the other group’s base pattern (a jpg of your grid drawing) on that group’s page. If you think that the other group’s instructions are invalid and you cannot complete the pattern, post your reasoning on that group’s page along with an image of your incomplete pattern.

NOTE: Groups will lose points in Week Two for not correctly following the base pattern instructions of another group or identifying why another group’s instructions are incomplete or invalid.
2.2. **Analysis and Reflection [20 Points]**

Post your Analysis and Reflection responses in the Discussion area of your page, NOT in the body of the page.

You are expected to discuss these analysis and reflection questions among your group. One member must start a new topic for EACH Analysis or Reflection by selecting “New Comment.” In the Topic area, type “Analysis” or “Reflection” and in the Comment area, paste in the Analysis or Reflection questions. Using the Analysis or Reflection questions as prompts, each member will post his or her responses as a reply to the original comment. This process will keep the group’s Analysis and Reflection in separate threads and make it easier to follow the development of your answers.

You will be graded individually based upon your contributions to the group Analysis or Reflection. In order to receive individual credit for Week 2, each group member must contribute to the answers to these questions. **Group members who do not contribute to the Analysis or Reflection Discussion will not receive points.**

**Analysis [10 points].** **Respond to these questions:**

1. Which operations created more interesting patterns or complex patterns that are more different from the base patterns? Explain.
2. Compare the number of lines of your instruction set for the complex pattern and the number of lines of your instruction set for the base pattern. Is it possible to reduce the number of lines in either set by using the three operations? If yes, please describe and show. If no, please explain.

**Reflection [10 points].** **Respond to these questions:**

1. Reflect on the impact of having a loop structure in your instruction set. What are the benefits or advantages for you as a designer of an instruction set, as well as for you as a “drawer” following an instruction set.
2. Reflect on the impact of having relative references in your instruction set on replicating or duplicating processes or solutions on different items. Illustrate your reflection using a recipe designed for 4 people being adapted for 8 people, for example.

---

**Deadlines and Hand-In**

Week 1 Deadline – [XXX, 11:59 p.m.]: You should have finished modifying your base pattern to use relative coordinates. You should also have posted the written instructions for your complex pattern using shift, reflect, and rotate in the body of your wiki page.

Week 2 Deadline – [XXX, 11:59 p.m.]: You should have attempted to follow the instructions of another group and posted your drawn solution to the other group’s wiki pages. Your individual Analysis and Reflection comments must be posted in the Discussion area of your group’s page.

---

**Grading**
Week 1. Group credit for generating the complex pattern instructions. Your instructions must meet the stated requirements and points are deducted for if the patterns do not use relative coordinates, do not contain rotating, reflecting and shifting elements or if your instructions are incomplete or invalid.

Week 2. Pattern Generation from another group’s instructions: Group credit for generating and posting another group’s pattern from their instructions OR for identifying the flaws in the other group’s instructions and posting the resulting incomplete pattern. Analysis and Reflection: graded individually. Each member must post in the Discussion with a minimum of 3-5 coherent, relevant sentences for full credit.

Late work will not be graded.