Create — Applications from Ideas
Written Response Submission Template

Submission Requirements

2. Written Responses

Submit one PDF document in which you respond directly to each prompt. Clearly label your responses **2a – 2d in order. Your response to all prompts combined must not exceed 750 words, exclusive of the Program Code.**

Program Purpose and Development

2a. Identify the programming language and identify the purpose of your program. Explain your video using one of the following:

- A written summary
- of what the video illustrates OR
- An audio narration in your video. If you choose this option, your response to the written summary should read, "The explanation is located in the video."

(Approximately 150 words)

Insert response for 2a in the text box below.

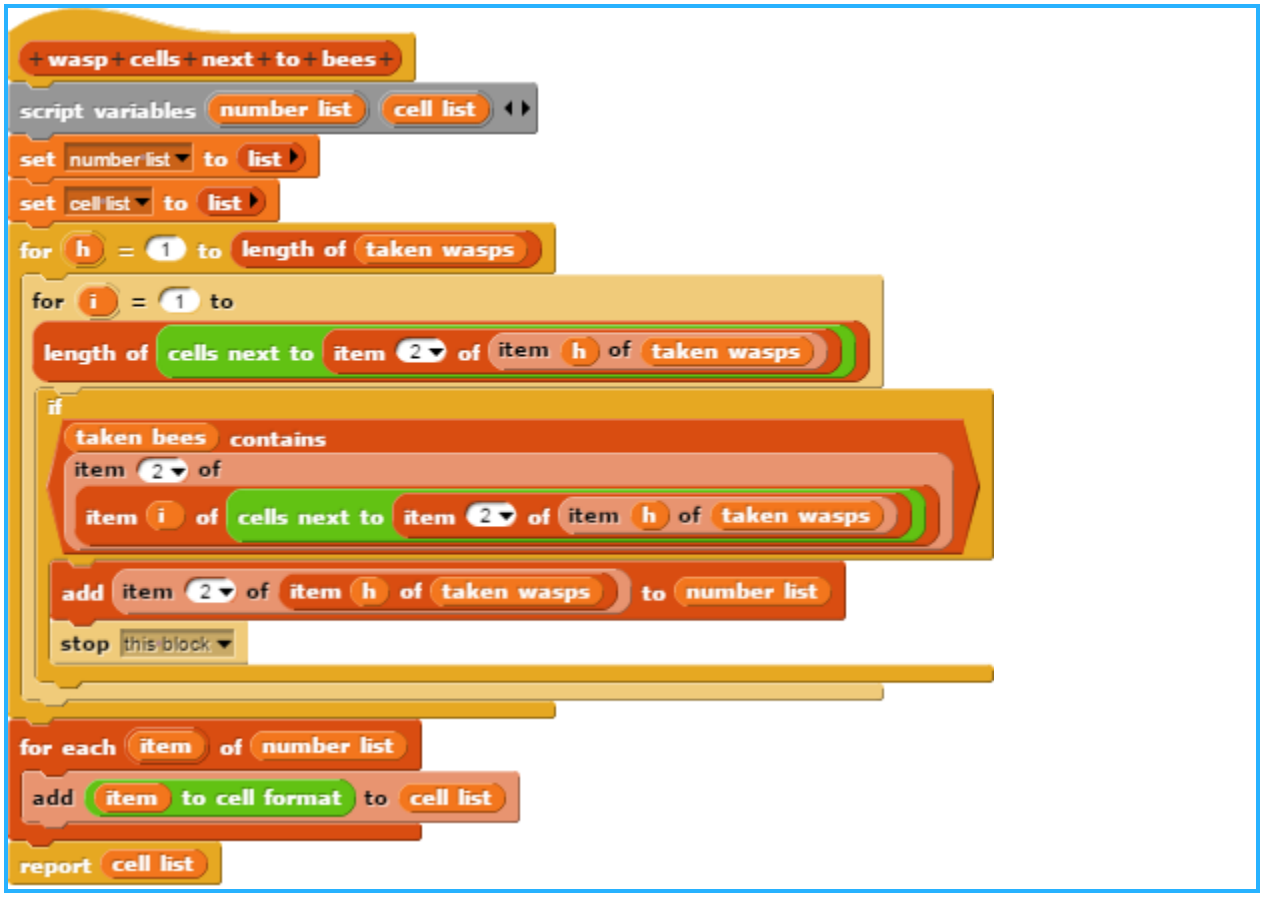
The program is a strategical domination game. The program was created in Snap! using the provided blocks to write the code. The purpose of the program is to have the user plan its placement and attacks to ultimately take over every cell and win the game. At the start, the user and computer alternate turns by placing a bee (user), or a wasp (computer). Once all the bees and wasps are placed, the user is prompted to choose a bee-occupied cell to attack from, and a wasp-occupied cell to attack. Here, the user and computer roll a die, and whoever rolls the highest score (number of bees/wasps in the cell + roll) wins. If the attacker wins, the attacker takes over that cell, replacing however many bees/wasps there were. The video shows the process of placing a bee, attacking a cell, rolling for score, and winning the game.

2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development; the second could refer to either collaborative or independent program development. *(Approximately 200 words)*

Insert response for 2b in the text box below.

There were many problems that arose while coding the program. One of the earlier problems encountered was developing a code to have the computer randomly, but correctly, place wasps in cells. This was a collaborative development. We had to make sure the computer would not place a wasp on top of another wasp. At first, we pondered the idea of having a large list of coordinates that the computer would accordingly follow. However, we agreed that that method would not be efficient and eventually decided to simply adjust the y-coordinate of the wasp so that additional wasps would be placed underneath any preexisting wasps. The second problem, an independent development, was developing a code to have the host determine a winner and accordingly replace the bees/wasps. In the code, there is a list variable for both the bees and wasps that keep track of which cells are occupied by whom. After some time experimenting with different options, I decided to go with a method similar to that of a selection sort algorithm. I wrote a code that swapped the two items, so that the cell would be added to the attacker's list and removed from the defender's list.

2c. Capture and paste an image or images of your program code segment that implements the most complex algorithm you wrote. (marked with a **color border** below)

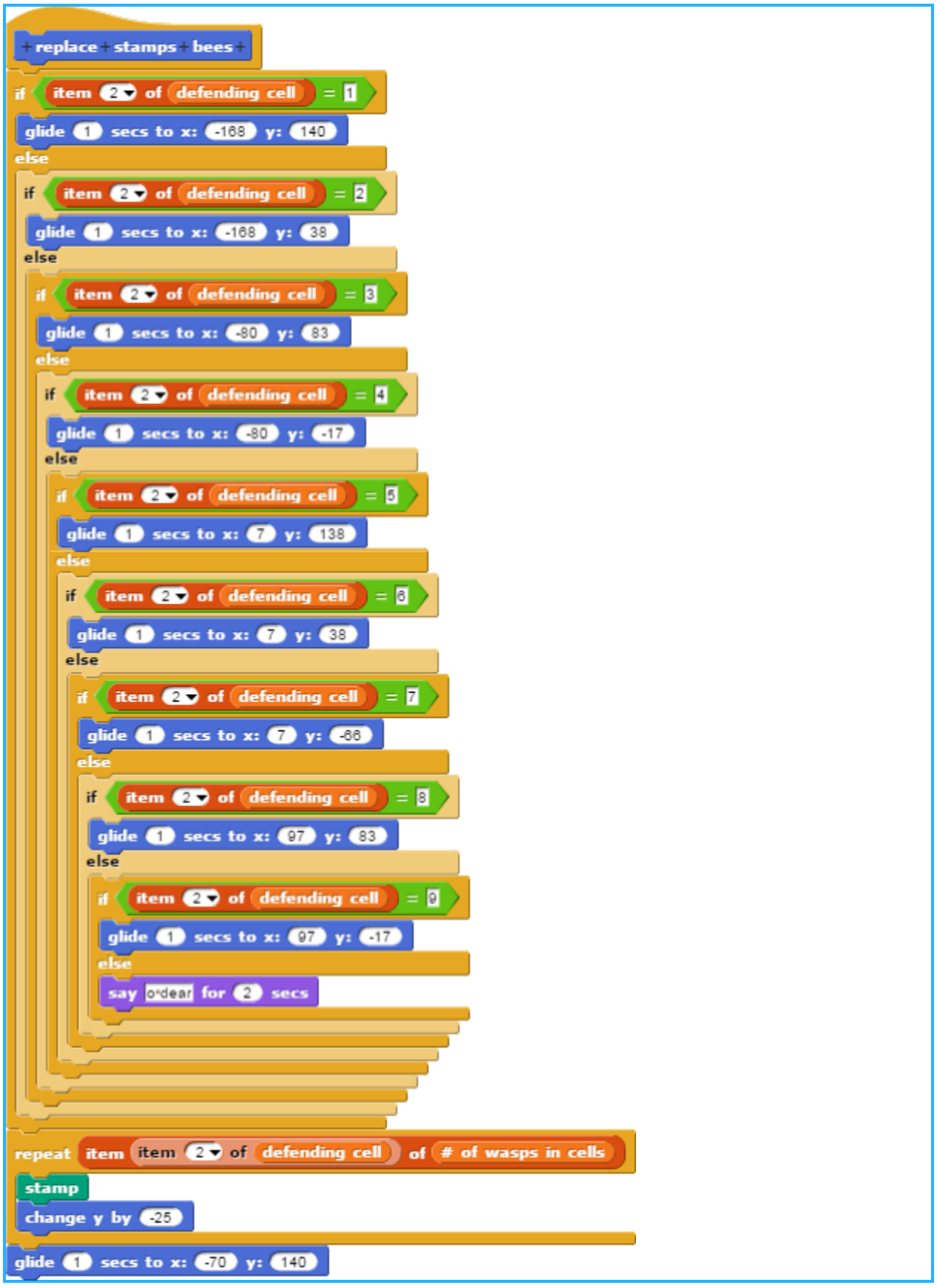


Your algorithm should integrate several mathematical and logical concepts. Describe the mathematical and logical concepts used to develop the algorithm. Explain the complexity of the algorithm and how it functions in the program. *(Approximately 200 words)*

Insert text response for 2c in the plain box below.

This particular algorithm is essential to the program because it assures that the computer will attack from a cell that is possible to attack from. If a certain wasp-occupied cell is bordering only wasp-occupied cell, that cell cannot attack a bee-occupied cell, since there are no neighboring bee-occupied cells to attack. Vaguely speaking, for each wasp-occupied cell, the algorithm checks if any bordering cells are bee-occupied and accordingly adds an appropriate cell to a list. This algorithm uses sequential search algorithms to attain its goal. The first sequential search, using the variable “h,” goes through the entire list of wasp-occupied cells. The second sequential search, using the variable “i,” searches the bordering cells of the current “h” cell and checks if any bordering cells are bee-occupied. If a bee-occupied cell is found to be neighboring the current “h” cell, then the current “h” cell is added to a list. Once the whole list of wasp-occupied cells has been checked, the final list is randomly chosen from to be the computer’s attacking cell.

2d. Capture and paste an image or images of the program code segment that contains an abstraction you developed (marked with a matching **blue color border** below)


+replace+stamps+bees+
if item 2 of defending cell = 1
glide 1 secs to x: -168 y: 140
else
if item 2 of defending cell = 2
glide 1 secs to x: -168 y: 38
else
if item 2 of defending cell = 3
glide 1 secs to x: -80 y: 83
else
if item 2 of defending cell = 4
glide 1 secs to x: -80 y: -17
else
if item 2 of defending cell = 5
glide 1 secs to x: 7 y: 138
else
if item 2 of defending cell = 6
glide 1 secs to x: 7 y: 38
else
if item 2 of defending cell = 7
glide 1 secs to x: 7 y: -66
else
if item 2 of defending cell = 8
glide 1 secs to x: 97 y: 83
else
if item 2 of defending cell = 9
glide 1 secs to x: 97 y: -17
else
say o*dear for 2 secs
repeat item item 2 of defending cell of # of wasps in cells
stamp
change y by -25
glide 1 secs to x: -70 y: 140

Your abstraction should integrate mathematical and logical concepts.
Explain how your abstraction helped manage the complexity of your program.
(Approximately 200 words)

Insert text response for 2d in the plain box below.

This particular abstraction is used in the process of replacing wasps with bees. It uses data stored about how many wasps were in the cell, and uses it to place bees in that cell. The abstraction uses mathematical coordinates to know where to place the bees such that no bees are overlapping. It also uses common logic to adjust its y-position if multiple bees are being placed so that additional bees are placed beneath the existing bees. This abstraction is vital to the program because it is used every time the user defeats the computer and takes over its cell. Similarly, a "replace stamps wasps" block was abstracted to do the same function as the "replace stamps bees" block, except it replaces bees with wasps. Both of these abstracted blocks help manage the complexity of the code by hiding the long list of coordinates and coding used to execute the tasks. The abstracted blocks both contain about twenty-five blocks of code, and by abstracting them, it keeps the main code clear and easy to debug in case of any problems.