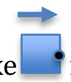
Part 1: Additional Documentation for Snake Game

1. Calculation and Geometry for Prey Capture

The program determines prey capture by checking for an overlap between the prey's rectangular boundaries and the snake's head portion. The boundaries are represented by their corner coordinates (top-left, top-right, bottom-left, bottom-right).

To find the corner coordinates of the prey we simply add the prey width to either the random x coordinate we generate, the random y, both, or neither. We treat neither as top left, just adding to x as top right, y as bottom left, and both as bottom right.

Finding the corner coordinates of the snake is a little trickier. The snake coordinate that we have is for the middle of the leading edge of snake.  This means that to find the corners we need to take into account the direction the snake is moving. Depending on the direction we add or subtract a different combination of snake width // 2 and snake width to x and y. This is because if we want to get to the left edge in our example we need to subtract snake width from x but if we want to get to the top edge we need to subtract snake width // 2 from y.

To check for prey capture we first check to see which is bigger snake width or prey width. Whichever is biggest will serve as our bounding area. We then take our smaller icon and check if any of its corners fall within the bounding area. We do this by checking if the corner's x and y are between the x's of the top left and bottom right corner of the bounding area and the y's. This checks if the corner coordinate is in the bounding area. We do this for all four corners and if there is overlap the prey is treated as consumed. This method works for reasonable size of snake or icon regardless of size difference

2. Calculation and Update of New Coordinates

The new head coordinates of the snake are calculated based on the current direction:
Left: Decrease the X-coordinate by the snake segment width.
Right: Increase the X-coordinate by the snake segment width.
Up: Decrease the Y-coordinate by the snake segment height.
Down: Increase the Y-coordinate by the snake segment height.

Snake Growth:
If prey is captured, the new head coordinates are appended to the list without removing the tail coordinates, resulting in the snake growing in length. If no prey is captured, the new head is appended, and the tail is removed, maintaining the snake's length.

3. Generation of New Prey

New prey coordinates are generated randomly while adhering to these constraints:
The prey must be at least a threshold distance (e.g., 15 px) away from the edges of the game window to avoid overlap.
The prey must not overlap with any part of the snake.

Avoiding the edge is done simply by adjusting parts of the range of the random integer generated for corner coordinates of the prey by the threshold distance and the width of the prey. (Note we chose to generate the corner rather than the middle coordinate for simplifying our hit detection and to allow for odd widths to still have integer corner coordinates)

To check if the prey is generated on the snake, we do a similar check as the prey capture. The difference is we find the corner coordinates of all snake width by snake width portions of the snake. To do this we loop through the list of snake coordinates checking to see what direction the snake moved in by comparing x and y coordinates. We calculate the corners the same way as in prey capture. We then loop through all the snake portions checking if there is overlap the same way as in prey capture. If there is overlap we generate a new prey coordinate and try again

4. Challenges Faced

It was a challenge to figure out the coordinate system. Originally, we were unsure how the tkinter line command treated the coordinates and didn't know where exactly our snake coordinates represented on the snake. After we figured out they were on the middle of the leading edge we were able to come up with a method to check for prey capture. Another challenge was not getting prey to generate on the snake. Our first implementation didn't take every single snake width by snake width portion of the snake into account. We at first grouped long segments into one big rectangle. This works fine if the snake width is already larger than the icon width to start, but if it is smaller to start, and then the new rectangle we grouped grows to be bigger the overlap checking fails to work. This has to do with checking the wrong corners vs bounding area. This took a while to figure out because all the math was correct and the prey only started generating incorrectly later in the game when the segments had grown long.

5. Incremental Improvements for the Future

Some improvements would be to optimize the method to check if generated prey overlaps any part of the snake. Checking every single portion of the snake works, but having it group together portions or disregarding some portions is a method that would work. We also would have liked to really figure out how tkinter treats bends in the line or non-integer widths/coordinates. For simplicity we just treated everything as truncated integers, but we could have gotten it precise.