**Memory V2 Reflection Activity**

**Q1. Describe how the game class determines when to stop the game, and list all the relevant code segments.**

|  |
| --- |
| it checks every tile to see if they are flipped. if all tiles are flipped, the game sets continue\_game to false  Code fragments:  stop=True  for row in self.tiles:  for tile in row:  if not tile.flipped:  stop=False  if stop:  self.continue\_game = False |

**Q2. Assume you also want to keep track in the game of how many times the user clicked anywhere in the window, independent of whether a tile is exposed or not. How do you have to change the code for the game class?**

|  |
| --- |
| 1) We would create a new variable called num\_clicked, and increment it every time the event type is mouse button up |

**Q3. Write a method in the Tile class that takes a string, denoting a color name such as ‘red’, ‘blue’, etc) as input and changes the border color of the tiles to that color.**

|  |
| --- |
| def change\_border\_color(self, color)  Tile.border\_color = pygame.Color(color) |

**Q4. Assume that you want to show a ticking “clock” *below* the score of the memory game. For this purpose, you have stored 60 images in an instance attribute of the game class self.clock\_images, which is a list with 60 elements, storing images of a “clock” with a second hand for each of the 60 seconds around the clock. The images should be shown in this sequence, changing the image that is drawn every second, over and over, starting with the 0th second, then the 1st second, and so on up to the 60th second, and then starting at be beginning again. Extend the draw\_score method accordingly (no change is necessary anywhere else in the code).**

|  |
| --- |
| def draw\_score(self):  size=60  image\_y=0  width=uaio.get\_width(str(self.score), size)  image\_x=self.surface.get\_width() - width  uaio.draw\_string(str(self.score), self.surface, (x, y), size)  add the following lines after the current code in draw\_score:    time=score % 59  image=time\_images[time]  self.surface.blit(image, (image\_x, image\_y)) |

**Q5. Consider the implementation of the select method in the Tile class. Re-write the method suite so that it consists of only a single return statement!**

|  |
| --- |
| def click(self, position)  return self.rectangle.collidepoint(position) |