```python
import pygame, random

#variable definitions#
fps = 120 # defining the framerate of the game (increasing will cause game to run faster)
clock, gameRun, prevTile = pygame.time.Clock(), True, -1
def reset(): #function that resets the game to start
    global gameSpeed, gameOver, tileList, points
    gameSpeed, gameOver, tileList, points = 2, False, [], 0
global gameSpeed, gameOver, tileList, points
reset()

#Screen Resolution#
sWidth, sHeight = 540, 960

#Inits#
pygame.init()
screen = pygame.display.set_mode((sWidth, sHeight))
scoreFont = pygame.font.Font("scoreFont.ttf", 100)

#Classes#
class Tile(): #class that creates a tile and all of its attributes
    def __init__(self):
        global prevTile
        self.pos = -240 #tile pos
        col = prevTile
        while col == prevTile:
            col = random.randint(0, 3) #randomly selects the collumn for the tile
        prevTile = col
        self.col = col #attribute storing the collumn location
        self.isClicked = False
        self.alpha = 0 #transparency of the tile
        self.trash = False

#Functions#
def renderBackG(): #renders the white background including the division lines
    pygame.draw.rect(screen, (255, 255, 255), (0, 0, 540, 960))
    for i in range(1, 4):
        pygame.draw.rect(screen, (100, 100, 100), (135*i, 0, 1, 960))

def delTiles(): #deletes any tiles that have already been completed and are off screen
    global tileList
    for i in range(len(tileList) - 1):
        if tileList[i].trash:
            del tileList[i]

def tileLoop(): #controls all the existing tiles
    global tileList, gameOver
    for i in range(len(tileList)): #traversing the list of all tiles
        if tileList[i].isClicked and tileList[i].alpha < 120:
            tileList[i].alpha += 5 #making the tile more transparent if it has been clicked
        pygame.draw.rect(screen, (tileList[i].alpha, tileList[i].alpha, tileList[i].alpha),
        (tileList[i].col*135, tileList[i].pos, 135, 240)) #draws the tile
        tileList[i].pos += gameSpeed
        if tileList[i].pos > 960:
            tileList[i].trash = True #makrs the tile to be deleted
        if tileList[i].pos > 840 and not tileList[i].isClicked:
            gameOver = True #calls the game to be reset

def renderTiles(): #creates each new tile
```

```
    global tileList
    if len(tileList) == 0:
        tileList.append(Tile()) #creates a new instance of the Tile class and adds it the tileList
    elif tileList[len(tileList) - 1].pos >= 0:
        tileList.append(Tile())
    tileLoop()
    delTiles() #removes all uneeded tiles

def keyLoop(keys, deathFlag): #checks if the tile is within bounds of the selection area and adds a
point if so.
    global tileList, points
    colFlag = False
    for i in range(len(keys)):
        if keys[i]:
            for j in range(len(tileList)):
                if tileList[j].col == i and tileList[j].pos > 580 and tileList[j].pos < 840: #checks
                pos of tile is within parameters
                    if tileList[j].isClicked == False:
                        points += 1
                    tileList[j].isClicked = True #sets itClicked attribute to true of tile
                    deathFlag, colFlag = False, False
                    break
                else:
                    deathFlag, colFlag = True, i*135
    return deathFlag, colFlag


def checkKeys(): #detects keyboard presses
    global tileList, gameOver, points
    deathFlag = False
    pressed = pygame.key.get_pressed() #gets list of all pressed keys
    keys = [pressed[pygame.K_a], pressed[pygame.K_s],
            pressed[pygame.K_k], pressed[pygame.K_l]]
    deathFlag, colFlag = keyLoop(keys, deathFlag) #sends keys pressed to keyLoop
    if deathFlag:
        gameOver = True
        pygame.draw.rect(screen, (255, 0, 0),   (colFlag, 0, 135, 960))

def renderScore(): #renders the score number at top of screen
    global gameSpeed
    score = scoreFont.render(str(points), True, (34, 136, 182)) #renders the score with the scoreFont
    from the points var
    scoreRect = score.get_rect() #creates, centers, and blits the score rectangle surface onto the
    screen
    scoreRect.center = (270, 100)
    screen.blit(score, scoreRect)
    if points <= 100:
        gameSpeed = (points / 20) + 3

def renderSelector(): #renders the selection bar at the bottom of the screen
    pygame.draw.rect(screen, (34, 136, 182), (0, 820, 540, 20))

#Main Loop#
while gameRun:
    clock.tick(fps) #makes sure that the game runs at the specified framerate by delaying
    for event in pygame.event.get():
        if event.type == pygame.QUIT: #checks if the x has been pressed
            gameRun = False
    renderBackG()
```

```
    renderTiles()
    checkKeys()
    renderSelector()
    renderScore()
    pygame.display.flip()
    if gameOver:
        reset() #calls reset to reset the game back the the beginning

#Close Game#
pygame.quit()
```