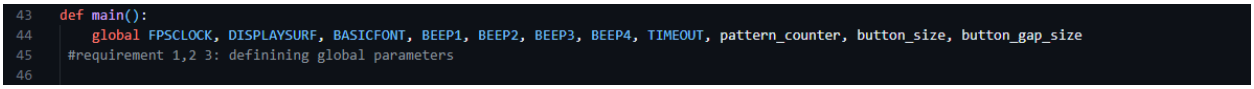
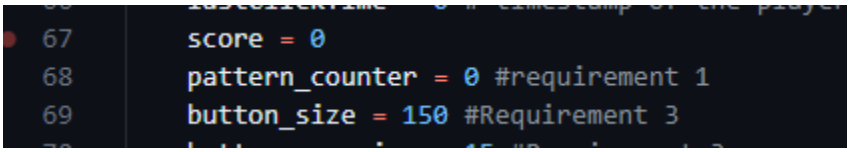
Лабораториска вежба Simulate – ПНВИ

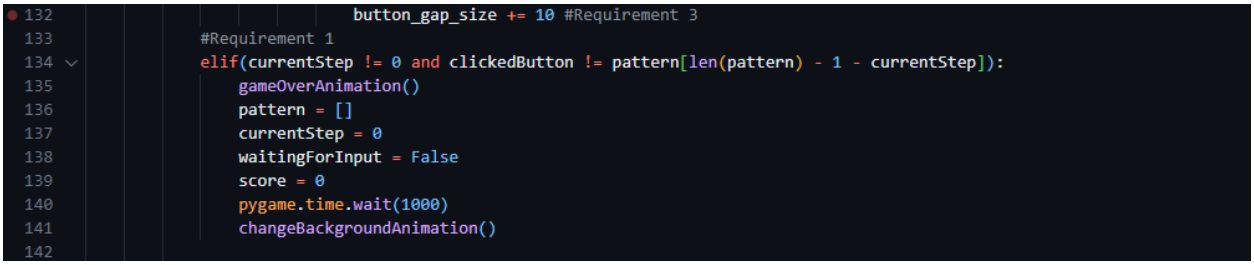
Фросина Цветковска 191216

1. Change the game so that the player must play back the sequence in reverse order.

За да може да го добиеме ефектот на reverse order, прво додаваме глобална променлива pattern_counter(линија 44) која што истата се сетира на 0 (линија 68). Во самиот main додаваме проверка во случај играчот да не го погоди првиот чекор или да згреши чекор во секвенцата да е потребно да ја игра секвенцата во обратен редослед.

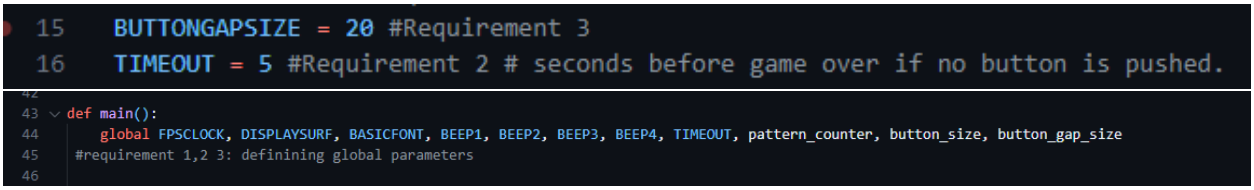


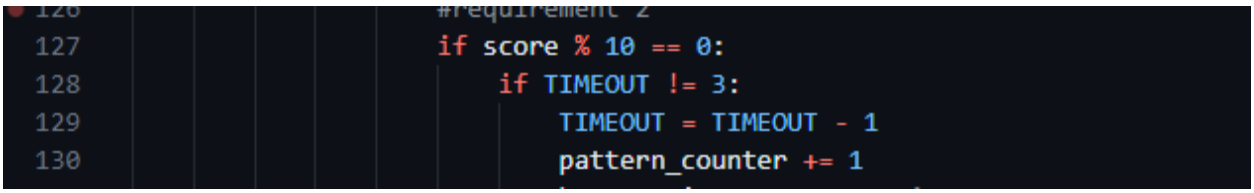




2. Decrease the value (starting from 5 to 3) of the timeout out for each 10 changes of the pattern.

Почнуваме со вредност од 5 за TIMEOUT (линија 16), мора да ја направиме променливата глобална ако сакаме да и ја менуваме вредноста (линија 44). Следно проверуваме ако играчот постигнал 10 точни повторувања на секвенцата, ја намалуваме вредноста на TIMEOUT за 1 (претходно проверувајќи TIMEOUT да не е еднаков на 3)(линии 127-129)





3. After the score of multiples of 10 is reached (10, 20, ...), increase the grid (the number of buttons by one for both dimensions, width and height).

На почеток додаваме вредности за BUTTONSIZE и BUTTONGAPSIZE и две глобални променливи button_size и button_gap_size. На секоја drawButtons() и flashButtonAnimation() се додаваат овие две вредности, но исто така во потписот на самата функција. Во flashButtonAnimation()

Додаваме во flashSurf димензиите за самите копчиња. Со цел да додаваме ново копче креираме нова функција createButtonRectangles која што со веќе дадените button-per_row и button_per_column ќе креира ново копче. Истото после секој 10ти резултат се зголемува за 1

```
14    BUTTONSIZE = 200 #Requirement 3
15    BUTTONGAPSIZE = 20 #Requirement 3
16    TIMEOUT = 5 #Requirement 2 # seconds before game over if no button is pushed.
17    button_size = BUTTONSIZE #Requirement 3
18    button_gap_size = BUTTONGAPSIZE #Requirement 3
19
```

```
43  def main():
44      global FPSCLOCK, DISPLAYSURF, BASICFONT, BEEP1, BEEP2, BEEP3, BEEP4, TIMEOUT, pattern_counter, button_size, button_gap_size, buttons_per_column, buttons_per_row
45  #requirement 1,2 3: defininig global parameters
46
```

```
74      while True: # main game loop
75          clickedButton = None # button that was clicked (set to YELLOW, RED, GREEN, or BLUE)
76          DISPLAYSURF.fill(bgColor)
77          drawButtons(button_size, button_gap_size, reverse=(pattern_counter % 2 == 1)) #Requirement 3
78
```

```
107             pattern.append(random.choice((YELLOW, BLUE, RED, GREEN)))
108             for button in pattern:
109                 flashButtonAnimation(button, button_size, button_gap_size, reverse=(pattern_counter % 2 == 1)) #Requirement 3
110                 pygame.time.wait(FLASHDELAY)
111             waitingForInput = True
```

```
125                     if score % 10 == 0:
126                         if TIMEOUT != 3:
127                             TIMEOUT = TIMEOUT - 1
128                             pattern_counter += 1
129                             button_size += 20 #Requirement 3
130                             button_gap_size += 10 #Requirement 3
131                             buttons_per_row += 1
132                             buttons_per_column += 1
133                             createButtonRectangles()
134
135                             for button in button_rectangles:
136                                 flashButtonAnimation(button, button_size, button_gap_size, reverse=(pattern_counter % 2 == 1))
137                                 pygame.time.wait(FLASHDELAY)
138             #Requirement 1
```

```
286 def createButtonRectangles():
287     global button_rectangles, XMARGIN, YMARGIN, BUTTONSIZE, BUTTONGAPSIZE, buttons_per_row, buttons_per_column
288     button_rectangles = [
289         pygame.Rect(XMARGIN + i * (BUTTONSIZE + BUTTONGAPSIZE),
290                     YMARGIN + j * (BUTTONSIZE + BUTTONGAPSIZE),
291                     BUTTONSIZE, BUTTONSIZE)
292         for j in range(buttons_per_column)
293         for i in range(buttons_per_row)
294     ]
295
```

```python
#Requirement 3: Added button_zide and button_gap_size
def flashButtonAnimation(color, button_size, button_gap_size, animationSpeed=50, reverse=False):
    if color == YELLOW:
        sound = BEEP1
        flashColor = BRIGHTYELLOW
        rectangle = YELLOWRECT
    elif color == BLUE:
        sound = BEEP2
        flashColor = BRIGHTBLUE
        rectangle = BLUERECT
    elif color == RED:
        sound = BEEP3
        flashColor = BRIGHTRED
        rectangle = REDRECT
    elif color == GREEN:
        sound = BEEP4
        flashColor = BRIGHTGREEN
        rectangle = GREENRECT

    origSurf = DISPLAYSURF.copy()
    flashSurf = pygame.Surface((button_size, button_size)) #Requirement 3
    flashSurf = flashSurf.convert_alpha()
    r, g, b = flashColor
    sound.play()
```

```python
#Requirement 3: Added button_size and button_gap_size
def drawButtons(reverse=False, button_size = BUTTONSIZE, button_gap_size = BUTTONGAPSIZE):
    if not reverse:
        pygame.draw.rect(DISPLAYSURF, YELLOW, YELLOWRECT)
        pygame.draw.rect(DISPLAYSURF, BLUE,   BLUERECT)
        pygame.draw.rect(DISPLAYSURF, RED,    REDRECT)
        pygame.draw.rect(DISPLAYSURF, GREEN,  GREENRECT)
    else:
        pygame.draw.rect(DISPLAYSURF, BRIGHTYELLOW, YELLOWRECT)
        pygame.draw.rect(DISPLAYSURF, BRIGHTBLUE,   BLUERECT)
        pygame.draw.rect(DISPLAYSURF, BRIGHTRED,    REDRECT)
        pygame.draw.rect(DISPLAYSURF, BRIGHTGREEN,  GREENRECT)

```

Attach the final code with marked and commented changes that you have made. Additionally, attach a document (PDF) where you will put images that will illustrate those changes, with a short explanation for the image and the change (what it is and how it was implemented). The attached solutions that will NOT contain the whole code and the explanations document, will NOT be considered.