ASSIGNMENT 2

Q1. Personal World Clock Class

Use the template code given here:
https://drive.google.com/file/d/1YYnZnIMvoBy1ZAGgHim0ev_C6YfXElk2/view?usp=sharing
Use the test file given here:
https://drive.google.com/file/d/1nD_YgZAqUpr9_4i9hud_brAtaah7Ztjf/view?usp=sharing

Q1. Personal World Clock Class

Write a personal world clock class named PersonalWorldClock. PS: Go through the template and test files. Complete explanation of what we expect from the class is added in the template file. Refer to the datetime library.

3 points

Q2. Binary Number Class

Write a class to represent binary numbers (base 2). Include a function to convert the number from binary to base 10 and print it to the screen.

3 points

```
class BinaryNumber():
    """Minimalist binary number class"""
    def __init__(self,a):


    def base_10(self):
        """Finds the numerical value (base 10) of the input"""

    def __str__(self):
        """Prints number in base 2 and base 10 notation to the screen with the base of number subscripted"""
```

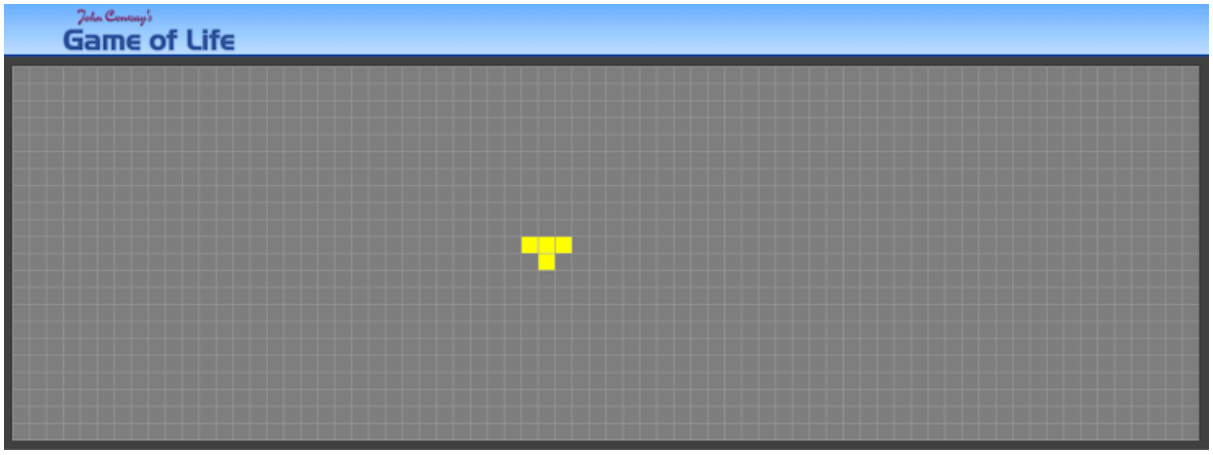
Q3. Conway's Game of Life

Use the template code given here:
https://drive.google.com/file/d/1NzBd4mYCoXBBPxFYNriv2MZhYcTLayW6/view?usp=sharing

Q3. Simulate Conway's Game of Life (starting at the configuration given below)

The universe of the Game of Life is an infinite, two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead, (or populated and unpopulated, respectively). Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur: Any live cell with fewer than two live neighbours dies, as if by underpopulation. Any live cell with two or three live neighbours lives on to the next generation. Any live cell with more than three live neighbours dies, as if by overpopulation. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction. Report the configuration obtained at iteration number 39.

5 points



Add file

Q4. Stacks

Write a Python code consisting of two functions "InfixToPostfix" and "EvaluatePostfix". "InfixToPostfix" should convert a given string representation of an infix expression to postfix (as a list of elements). "EvaluatePostfix" should compute and return the numerical value of a postfix expression. Please return the required output in both functions. Assume that the only possible operators are '+, -, /, *, ^' (^ is the exponent operator). Make sure to include documentation for each function. Please refer the image below for more information. NOTE: For simplicity, the numeric entities in the infix expression are positive integers and not negative or floating point numbers.

3 points

```
infix = "3^4/(5*6)+10"
postfix = InfixToPostfix(infix)
print("Postfix expression: ", postfix)
result = EvaluatePostfix(postfix)
print("Value of expression: ",result)
```

```
Postfix expression:  [3, 4, '^', 5, 6, '*', '/', 10, '+']
Value of expression:  12.7
```

Q5. Stacks and Queues

The template code for this question is given here: https://drive.google.com/file/d/1K1CLZZ9Os4ADQz6YKIae1GERInqxZgS8/view?usp=sharing

Q5. Stacks and Queues

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches. The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack. At each step: If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue. Otherwise, they will leave it and go to the queue's end. This continues until none

of the queue students want to take the top sandwich and are thus unable to eat. You are given two integer arrays students and sandwiches where sandwiches[i] is the type of the ith sandwich in the stack (i = 0 is the top of the stack) and students[j] is the preference of the jth student in the initial queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

6 points

Example 1:

```
Input: students = [1,1,0,0], sandwiches = [0,1,0,1]
Output: 0
Explanation:
- Front student leaves the top sandwich and returns to the end of the line making students =
[1,0,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students =
[0,0,1,1].
- Front student takes the top sandwich and leaves the line making students = [0,1,1] and sandwiches =
[1,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [1,1,0].
- Front student takes the top sandwich and leaves the line making students = [1,0] and sandwiches =
[0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [0,1].
- Front student takes the top sandwich and leaves the line making students = [1] and sandwiches = [1].
- Front student takes the top sandwich and leaves the line making students = [] and sandwiches = [].
Hence all students are able to eat.
```

Example 2:

```
Input: students = [1,1,1,0,0,1], sandwiches = [1,0,0,0,1,1]
Output: 3
```