THE UNIVERSITY OF THE WEST INDIES
Department of Computing
COMP1127–Introduction to Computing II

Lab 2

Exercise

Type these expressions in the python shell.

```
>>> [] + [1,2]

>>> [1,2]+[3]

>>> len(['apples','oranges',True,4])

>>> [x for x in range (0,10) if x % 3 == 0 ]

>>> [x for x in range (0,10) if x % 2 == 0 or x % 5 == 0]

>>> len([x for x in range (0,10) if x % 2 == 0])

>>> [(x,y,2010) for x in range (1,8) for y in range(1,13)]
```

Access and Complete the following Lab Exercise on Hackerrank.

Opening Date: **Friday, October 28, 2022**

Lab Date (Week 2): **Monday, October 31 – Saturday, November 5, 2022**
Due Date: **11:45 pm, Sunday, November 6, 2022 (on Hackerrank)**

Problem 1

Include the following list `month_days`.

```
month_days = [('January',[31]),('February',[28,29]),('March',[31]),
('April',[30]),('May',[31]),('June',[30]),('July',[31]),('August',[31]),
('September',[30]),('October',[31]),('November',[30]),('December',[31])]
```

Write a function `days_in_month` which takes a month as an argument and finds the corresponding month in the `month_days` list and returns the list with the number of days associated with that month. Where the given month name is not found, return the empty list.

```
>>> days_in_month('December')
[31]
>>> days_in_month('February')
[28, 29]
```

Problem 2

Include the following `day_names` in your code.

```
day_names =
['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday']
```

Zeller's Congruence is an algorithm for finding the day of the week for any date. Zeller's formula is as follows:

day = (((13*m+3) / 5 + d + y + (y / 4) - (y / 100) + (y / 400)) %7)

where
d = day, y = year and m = month

Note: If the month is January or February then you add 12 to the month and subtract 1 from the year before calculating the day. Further, Zeller's Congruence is deemed to be computed correctly where the integer value is used for each computation of a division (or number being divided).

The result is a day number in the range 0..6 where the corresponding day can be extracted from the `day_names` list by using an appropriate index.

e.g. `day_names[0] = 'Monday' and day_names[6] = 'Sunday'.`

Define a python function `day_of_week`, which displays the day name for a given date supplied in the form (day,month,year).
e.g.

```
>>> day_of_week(9,5,2010)
'Sunday'
>>> day_of_week(23,1,2010)
'Saturday'
>>> day_of_week(23,2,2010)
'Tuesday'
```

Problem 3

Using list comprehension, define a python function `unlucky`, which returns all the days in a given year which have the date Friday 13th
e.g.

```
>>> unlucky(2010)
[(13, 8, 2010)]
>>> unlucky(2009)
[(13, 2, 2009), (13, 3, 2009), (13, 11, 2009)]
```

[Hint: you need two ranges one for day starting from 1 and going to 31 and another one for month starting from 1 going to 12. Using these and the year which comes as an argument and use the function `day_of_week` (Problem 2) in the if part of list comprehension to check if a given date is 'Friday' and also check if the day is equal to 13.]

Problem 4

Write a python function `mostUnlucky`, which lists all the years between a given start year and end year e.g. years 0 and 2010 which have 3 unlucky days. Use function `unlucky` (Problem 3) to get a list of unlucky dates for a particular year and find the length of this list. If the length is greater than 2 then the year is added to another list which is returned as output.
e.g.

```
>>> mostUnlucky(1000,2018)
[1001, 1004, 1007, 1018, 1029, 1032, 1035, 1046, 1057, 1060,
1063, 1074, 1085, 1088, 1091, 1103, 1114, 1125, 1128, 1131, 1142,
1153, 1156, 1159, 1170, 1181, 1184, 1187, 1198, 1209, 1212, 1215,
1226, 1237, 1240, 1243, 1254, 1265, 1268, 1271, 1282, 1293, 1296,
1299, 1305, 1308, 1311, 1322, 1333, 1336, 1339, 1350, 1361, 1364,
1367, 1378, 1389, 1392, 1395, 1401, 1404, 1407, 1418, 1429, 1432,
1435, 1446, 1457, 1460, 1463, 1474, 1485, 1488, 1491, 1503, 1514,
1525, 1528, 1531, 1542, 1553, 1556, 1559, 1570, 1581, 1584, 1587,
1598, 1609, 1612, 1615, 1626, 1637, 1640, 1643, 1654, 1665, 1668,
1671, 1682, 1693, 1696, 1699, 1705, 1708, 1711, 1722, 1733, 1736,
1739, 1750, 1761, 1764, 1767, 1778, 1789, 1792, 1795, 1801, 1804,
1807, 1818, 1829, 1832, 1835, 1846, 1857, 1860, 1863, 1874, 1885,
1888, 1891, 1903, 1914, 1925, 1928, 1931, 1942, 1953, 1956, 1959,
1970, 1981, 1984, 1987, 1998, 2009, 2012, 2015]
>>>
```