ASSIGNMENT No. 3
Complex Computing Problem

(CLO3,PLO4,C5) Question#1:

Regular expressions are one of the more useful tools for many NLP tasks. The goals of this assignment are to familiarize yourself with regular expressions, to play with them in different languages/environments, to get some experience with some useful tools and to get a feeling for text data analysis. This assignment involves coding and experimentation. You have to write a function for each task that takes as input a string and returns a Boolean indicating whether that string is valid or invalid. Any programming language and/or platform (python, java, c++ etc.) and libraries can be used for coding the following tasks.

a) **Design** a regular expression that matches the strings that are valid identifiers such that each word should start with an alphabet then any combination of alphabet and digits or of special character and digits. Alphabet can be in upper or lower case, special characters include underscore _, $, % and @ symbols. a_b123, x, x$1 are all valid identifiers while %x, 123xyz, _x, $abc are not valid.

```python
import re

def starts_with_alphabet(prompt):
    # [A-Z|a-z] for the first character
    # [A-Z|a-z|0-9|_|,|$|%|@]* for infinite amount of given characters
following the first one
    temp = re.match("^[A-Z|a-z][A-Z|a-z|0-9|_|,|$|%|@]*$", prompt)

    if temp is not None:
        return True
    else:
        return False

if __name__ == '__main__':

    prompt = "b038tu03wofhwodivujpwo"
    print(prompt, starts_with_alphabet(prompt))
    prompt = "038tu03wofhwodivujpwo"
    print(prompt, starts_with_alphabet(prompt))
    prompt = "*038tu03wofhwodivujpwo"
    print(prompt, starts_with_alphabet(prompt))
```

```
Run: parta
D:\Anaconda\python.exe C:\Users\sm\PycharmProjects\ToA_CCP\
b038tu03wofhwodivujpwo True
038tu03wofhwodivujpwo False
*038tu03wofhwodivujpwo False

Process finished with exit code 0
```

b) **Design** the regular expression date, which only matches dates written as ten-character strings YYYY-MM-DD. For example, 2023-04-01 represents April 1 2023, whereas 2023 04-31 is not a valid date. You should account for leap years by assume that Feb 29th is valid if the year is evenly divisible by four. E.g., 2024-02-29 is a valid date but 2023-02-29 is invalid.

```
import re
def date_matching(prompt):
    # Check that the prompt is in the yyyy-mm-dd format
    odd_months = re.match("^[0-9]{4}-([0][1-9]|[1][0-2])-([0-2][0-9]|[3][0-
1])$", prompt)
    even_months = odd_months

    leap_year = False
    is_feb = False

    # Return False if prompt is NOT in the yyyy-mm-dd format
    if odd_months is None:
        return False

    # Extract year and check if leap year
    year = int(prompt[:4])
    leap_year = year % 4 == 0

    # Check if the month is February
    is_feb = int(prompt[5:7]) == 2

    if is_feb:
        if leap_year:
            # Allow 29 in date
            even_months = re.match("^[0-9]{4}-([0][2])-([0][1-9]|[1,2][0-
9])$", prompt)
        else:
            # Only allow up til 28 in date
            even_months = re.match("^[0-9]{4}-([0][2])-([0][1-9]|[1][0-
9]|[2][0-8])$", prompt)

    else:
        # Check for even months
        even_months = re.match("^[0-9]{4}-([0][4,6,9]|[1][1])-([0][1-
9]|[1,2][0-9]|[3][0])$", prompt)
        # Check for odd months
        odd_months = re.match("^[0-9]{4}-([0][1,3,5,7,8]|[1][0,2])-([0][1-
9]|[1,2][0-9]|[3][0-1])$", prompt)

    # Check if there is a match in any of the combinations tested i.e not
None
    if (even_months is not None) or (odd_months is not None):
        return True
    else:
        return False
```

```python
if __name__ == '__main__':

    prompt = "2023-04-31"
    print(prompt, date_matching(prompt))
    prompt = "2023-04-30"
    print(prompt, date_matching(prompt))
    prompt = "2024-02-29"
    print(prompt, date_matching(prompt))
    prompt = "2023-02-28"
    print(prompt, date_matching(prompt))
```

```
Run: partb
D:\Anaconda\python.exe C:\Users\sm\PycharmProjects\ToA_CCP\.idea\partb.py
2023-04-31 False
2023-04-30 True
2024-02-29 True
2023-02-28 True

Process finished with exit code 0
```

c) **Develop** the regular expression phone, which only matches mobile phone numbers in the format 03XZ-YYYYYYY for mobile numbers in Pakistan, where the letter X is the single letter code assigned to a specific mobile telephone operator and Z-YYYYYYY is the local telephone number from any mobile phone or Land Line.

- 3 - is the Mobile Access code
- Z can be any value between 0 and 9, assigned by the operator itself,
- X=0 Jazz Pakistan
- X=1 Zong Pakistan
- X=2 Jazz Pakistan
- X=3 Ufone
- X=4 Telenor Pakistan
- X=5 SCOM (Z=5 only)
- X=6 Instaphone

Existing Codes

- 300, 301, 302, 303, 304, 305, 306, 307, 308, 309 - Jazz Pakistan
- 310, 311, 312, 313, 314, 315, 316, 317, 318, 319 - Zong Pakistan
- 320, 321, 322, 323, 324, 325, 326, 327, 328, 329 - Jazz Pakistan
- 330, 331, 332, 333, 334, 335, 336, 337, 338, 339 - Ufone

- 340, 341, 342, 343, 344, 345, 346, 347, 348, 349 - Telenor Pakistan
- 355 - SCOM
- 364 – Instaphone

International callers must dial +92-3XZ-YYYYYYY to reach a mobile number in Pakistan from outside Pakistan, where '+92' is the Country Code, '3XZ' is Mobile Access Code as per the above list and 'YYYYYYY' is personal number.

Within Pakistan the same number can be reached by dialing either 03XZ-YYYYYYY or '0092-3XZ-YYYYYYY' or '+92-3XZ-YYYYYYY' from any mobile device or land line.
For example 0092-333-1234567, 0333-1234567 are valid but 0399-1234567,0123-1234567 are invalid.

```
import re
def check_phone_number(prompt):
    offset_index = 0
    # General phone number format
    exp = r"3[0-9]{2}-[0-9]{7}$"

    # 3 Checks if phone number is in valid format
    ##############################################################
    valid1 = re.match(r"^0092-" + exp, prompt)
    # Plus (+) is a special character in regex so we add '\' before
    valid2 = re.match(r"^\+92-" + exp, prompt)
    valid3 = re.match(r"^0" + exp, prompt)
    ##############################################################

    # If prompt does not match any pattern, it is in an invalid input
    if (valid1 is None) and (valid2 is None) and (valid3 is None):
        print("Given number is invalid.")
        return False

    if valid1 is not None:
        # 0092-312-1234567
        # ______^
        offset_index = 6
    elif valid2 is not None:
        # +92-312-1234567
        # _____^
        offset_index = 5
    elif valid3 is not None:
        # 0312-1234567
        # __^
        offset_index = 2

    # Dictionary of telcos and their keys
    telecoms = {0: 'Jazz', 1: 'Zong', 2: 'Jazz', 3: 'Ufone', 4: 'Telenor',
5: 'SCOM', 6: 'Instaphone'}
```

```
    # Extract the key from the prompt for identifying which telecom
    X = int(prompt[offset_index])
    # Extract the key from the prompt for identifying which code within the
telecom is used
    Z = int(prompt[offset_index + 1])

    # Get the name of the telecom from the dictionary
    operator = telecoms[X]

    # Check if it is a valid SCOM number
    if X == 5 and Z != 5:
        print("Given number is invalid.")
        return False

    # Check if it is a valid Instaphone number
    elif X == 6 and Z != 4:
        print("Given number is invalid.")
        return False

    print(f"This is a {operator} number and is valid.")
    return True

if __name__ == '__main__':

    prompt = "+92-326-1234567"
    print(prompt, check_phone_number(prompt))
    prompt = "+92-355-1234567"
    print(prompt, check_phone_number(prompt))
    prompt = "+92-365-1234567"
    print(prompt, check_phone_number(prompt))
```

```
Run: script
D:\Anaconda\python.exe C:\Users\sm\PycharmProjects\ToA_CCP\
This is a Jazz number and is valid.
+92-326-1234567 True
This is a SCOM number and is valid.
+92-355-1234567 True
Given number is invalid.
+92-365-1234567 False
```