Task2: Regex

Q1) https://www.hackerrank.com/challenges/introduction-to-regex?isFullScreen=true

```
import re
N = int(input())
for _ in range(N):
    print(re.search(r'^([-\+])?\d*\.\d+$', input()) is not None)
```

Q2) https://www.hackerrank.com/challenges/re-split?isFullScreen=true

```
import re
regex_pattern =  r"[.,]+"
print("\n".join(re.split(regex_pattern, input())))
```

Q3) https://www.hackerrank.com/challenges/re-group-groups?isFullScreen=true

```
import re
S = re.findall(r"([A-Za-z0-9])\1+",input("S: "))
if S:
    print(S[0])
else:
    print(-1)
```

Q4) https://www.hackerrank.com/challenges/re-findall-re-finditer?isFullScreen=true

```
import re

s = input("S: ")
lst = re.findall(r'(?<=[QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm])[aeiouAEIOU]{2,}(?=[QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm])', s)

if len(lst) > 0:
    for element in lst:
        print(element)
else:
    print(-1)
```

Q5) https://www.hackerrank.com/challenges/re-start-re-end?isFullScreen=true

```
import re
S = input("S: ")
k = input("k: ")
pattern = re.compile(k)
r = pattern.search(S)
if not r: print("(-1, -1)")
while r:
    print("({0}, {1})".format(r.start(), r.end() - 1))
    r = pattern.search(S,r.start() + 1)
```

Q6) https://www.hackerrank.com/challenges/re-sub-regex-substitution?isFullScreen=true

```
import re

N = int(input("N: "))

for i in range(0,N):
   txt = input()
   txt = re.sub(r"\ \&\&\ "," and ",txt)
   txt = re.sub(r"\ \|\|\ "," or ",txt)
   txt = re.sub(r"\ \&\&\ "," and ",txt)
   txt = re.sub(r"\ \|\|\ "," or ",txt)
   print(txt)
```

Q7) https://www.hackerrank.com/challenges/validate-a-roman-number?isFullScreen=true

```
 import re
thousand = 'M{0,3}'
hundred = '(C[MD]|D?C{0,3})'
ten = '(X[CL]|L?X{0,3})'
digit = '(I[VX]|V?I{0,3})'
regex_pattern = r"%s%s%s%s$" % (thousand, hundred, ten, digit)
print(str(bool(re.match(regex_pattern, input()))))
```

Q8) https://www.hackerrank.com/challenges/validating-the-phone-number?isFullScreen=true

```
import re
N = int(input())
for _ in range(N):
    if re.match(r'[789]\d{9}$',input()):
        print('YES')
    else:
        print('NO')
```

Q9) https://www.hackerrank.com/challenges/validating-named-email-addresses?isFullScreen=true

```
import re

N = int(input())

for i in range(N):
    name, email = input().split()
    pattern="<[a-z][a-zA-Z0-9\-\.\_]+@[a-zA-Z]+\.[a-zA-Z]{1,3}>"
    if bool(re.match(pattern, email)):
        print(name,email)
```

Q10) https://www.hackerrank.com/challenges/hex-color-code?isFullScreen=true

```
import re
N=int(input("N: "))

for i in range(0,N):
    s=input()

    x=s.split()

    if len(x)>1  and  '{' not in x:
        x=re.findall(r'#[a-fA-F0-9]{3,6}',s)
        [print(i) for  i in x]
```

Q11) https://www.hackerrank.com/challenges/html-parser-part-1?isFullScreen=true

```
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print ('Start :',tag)
        for ele in attrs:
            print ('->',ele[0],'>',ele[1])

    def handle_endtag(self, tag):
        print ('End   :',tag)

    def handle_startendtag(self, tag, attrs):
        print ('Empty :',tag)
        for ele in attrs:
            print ('->',ele[0],'>',ele[1])

MyParser = MyHTMLParser()
MyParser.feed(''.join([input().strip() for _ in range(int(input()))]))
```

Q12) https://www.hackerrank.com/challenges/html-parser-part-2?isFullScreen=true

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):


    def handle_comment(self, data):
        if (len(data.split('\n')) != 1):
            print(">>> Multi-line Comment")
```

```
            else:
                print(">>> Single-line Comment")
            print(data.replace("\r", "\n"))
        def handle_data(self, data):
            if data.strip():
                print(">>> Data")
                print(data)


html = ""
for i in range(int(input())):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

Q13) https://www.hackerrank.com/challenges/detect-html-tags-attributes-and-attribute-values?isFullScreen=true

```
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(tag)
        [print('-> {} > {}'.format(*attr)) for attr in attrs]

html = '\n'.join([input() for _ in range(int(input()))])
parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

Q14) https://www.hackerrank.com/challenges/validating-uid?isFullScreen=true

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re

for i in range(int(input())):
    N = input().strip()
    if N.isalnum() and len(N) == 10:
        if bool(re.search(r'(.*[A-Z]){2,}',N)) and bool(re.search(r'(.*[0-9]){3,}',N)):
            if re.search(r'.*(.).*\1+.*',N):
                print('Invalid')
            else:
                print('Valid')
        else:
            print('Invalid')
    else:
        print('Invalid')
```

Q15) https://www.hackerrank.com/challenges/validating-credit-card-number?isFullScreen=true

```
import re
pattern = re.compile(
    r'^'
    r'(?!.*(\d)(-?\1){3})'
    r'[456]\d{3}'
    r'(?:-?\d{4}){3}'
    r'$')

for _ in range(int(input().strip())):
   print('Valid' if pattern.search(input().strip()) else 'Invalid')
```

Q16) https://www.hackerrank.com/challenges/validating-postalcode?isFullScreen=true

```
import re

P = input()

regex_integer_in_range = r'^[1-9][\d]{5}$'
regex_alternating_repetitive_digit_pair = r'(\d)(?=\d\1)'

print (bool(re.match(regex_integer_in_range, P))
```

```
        and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

Q17) https://www.hackerrank.com/challenges/matrix-script?isFullScreen=true

```
import math
import os
import random
import re
import sys

first_multiple_input = input().rstrip().split()
n = int(first_multiple_input[0])
m = int(first_multiple_input[1])
matrix = []
for _ in range(n):
    matrix_item = input()
    matrix.append(matrix_item)
encoded_string = "".join([matrix[j][i] for i in range(m) for j in range(n)])
pat = r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])'
print(re.sub(pat,' ',encoded_string))
```