Home Work № 1

CSCI B505 Applied Algorithms 09/02/2022

Instructions

1. Under no circumstance the deadline will NOT be extended
2. Code should be written in python language in a Jupyter notebook
3. Upload Jupyter notebook and html file of the Jupyter notebook in canvas. No other forms submissions is considered as valid submission. Make sure to have the output of the required cells of the jupyter notebook and its html version before making submission.
4. Plagiarism is unacceptable and we have ways to find it. So do not do it.
5. There are six problems. Problem 1 - 20 points Problem 2, Problem 3 and Problem 4, Problem 5 - 15 points, Problem 6 - 20 points.
6. The code should readable with variables named meaningfully
7. Write test cases wherever required so that they cover all scenarios.

Problem 1

Olivia is always fascinated by prime numbers. She wanted to know the number of English letters that are required for writing them.
Write a program which takes the natural number N and gives the number of english letters required to write the prime numbers below N.

```
Constraint:  N < 100

Example 1:

Input N = 8
The prime numbers below 8 are 2 , 3, 5 and 7
2 - two - 3
3 - three - 5
5 - five - 4
7 - seven - 5
3 + 5 + 4 + 5 = 17
Output = 17
```

```
Example 2:

Input N = 3
The prime numbers below 3 are just 2
2 - two - 3
Output = 3
```

```
def count_prime_letters(n):
    #***logic****#

    #***********#
    return count
```

Problem 2

A perfect square is a number that can be expressed as a product of an integer with itself. (For example, 49 is the product of 7 with itself).A function $F(N)$ is defined such that it returns the sum of all perfect squares less than or equal to N, where N is a positive integer. An advanced version of $F(N)$, function $get\_square\_sum(M, N)$, is defined such that it applies operation $F(N)$ 'M' times to the result of the previous operation, starting from N.
Write the function $get\_square\_sum(M, N)$.You may assume $M >= 1, N >= 1$.

```
Example 1:
    get_square_sum(1,10):
    here M=1, N=10
    Sum of all perfect squares <=10 is:
    1 + 4 + 9 = 14
    Since M=1, we will stop here.
    output = 14

Example 2:
    get_square_sum(2,20):
    here M=2, N=20
    Sum of all perfect squares <=20 is:
    1 + 4 + 9 + 16 = 30
    Sum of all perfect squares <=30 is:
    1 + 4 + 9 + 16 + 25 = 55
    Since M=2, we will stop here.
    output = 55
```

```
1
2 def get_square_sum(M,N):
3     #***logic****#
4
5     #************#
6     return output
```

Problem 3

Every integer can be represented in binary notation, which is a string of 1s and 0s.

For example,

2 = 10

7 = 111

8 = 1000

You can explore this in python by using the python function ‘bin‘.

Given an Integer N. The task is to find the second set bit position and return the position. If an Integer is not having the second bit set, return -1. The second set bit position is the index of the second 1 in the binary representation of a number when viewed from right to left.

Note: Position count starts from 1.

Example 1:

N = 2

Binary representation of 2 is 10

Output = -1

Example 2:

N = 3

Binary representation of 3 is 11

Output = 2

Example 3:

N = 8

Output = -1

```
Example 4:
N = 9
Binary representation of 9 is 1001
Output = 4

Example 5:

N = 31
Binary representation of 31 is 11111
Output = 2
```

```
1    def check_bit(N):
2        #***logic****#
3
4         #************#
5        return (position of second bit )
```

Problem 4

Given a 3 digit prime number n, print an integer that is perfectly divisible by 7, 11, 13 and n. Accomplish the above task without using arithmetic operator '*' or any kind of loop or recursion, and clearly explain your solution in the comments.

```
1    def find_multiple(n):
2        # Your logic goes here
3
4       return the desired integer
```

Problem 5

There are 'n' number of teams participating in a football tournament. In every match the winning team is awarded 3 points while the losing team receives 0 points.
In case of a tie, both the teams receive 1 point each. Given three arrays $matches\_won, matches\_drawn$ and $goals\_scored$ each of length 'n' with value at $matches\_won[i]$ indicating number of matches the ith team won, $matches\_drawn[i]$ indicating the number of matches played by ith team that resulted in a tie and $goals\_scored[i]$ indicating the number of goals scored by the ith team in the entire tournament([0,n)).

The team with the most number of points is declared the champions of the tournament. In case two or more teams score the same number of points the team with the highest number of goals scored in the tournament is declared the winner. Assuming there is always a unique winner, return the index of the winning team.

```
Example:
Input:
    matches_won = [3,1,2,0]
    matches_drawn = [0,1,0,1]
    goals_scored=[6,4,5,2]
Output: 0
```

```
1  def find_the_winner(matches_won, matches_drawn, goals_scored):
2      # Your logic goes here
3
4      # return the 0-based index of the winning team
```

Problem 6

Mr.Jack spent his last weekend busy decoding the cipher to open the lock of a treasure.
He was able to decode at the end but figured out from Mr. Smith later that the decoded string wouldn't be valid as there are some extra characters added into the string.
Mr.Smith knew that the characters which were added to the string are the 3rd occurrences of the character. Could you help Mr.Jack to figure out the cipher.

```
Examples:

    S = ‘‘abzzbabz’’ (’z’ and ’b’ are repeated 3 times in the sequence
    so we need to remove the 3rd occurence of both the characters)
    O/p - ‘‘abzzba’’

    S = ‘‘abzzbabzz’’ (’z’ is repeated 4 times and ’b’ is repeated 3 times
    so we need to remove only the 3rd occurence of both the characters)
    O/p - ‘‘abzzbaz’’

    S = ‘‘aaaaaa’’  (’a’ is repeated 6 times
    so every 3rd occurence of ’a’ is removed from the given cipher)
    O/p - ‘‘aaaa’’

Constraints:
    0<= S <= 10^6
```

```
Expected Time Complexity - O(n)
Expected Space Complexity - O(n)
```

Given a cipher, return the decoded string?

```
1    def decoded_String(S):
2        #***logic****#
3
4         #***********#
5    return (decoded_String)
```