This homework asks you to write a python function and answer questions based on the program. You should submit the source code of the program and the report together. Since the report needs substantial work, you should first complete the program and test it thoroughly as soon as possible.

1. Write a function named *factors* that returns all prime factors of an integer. For example, factors(12) returns [2,2,3]. If the input is a prime or 1 it returns an empty list. The factors should be listed in increasing order.

Please do not search or copy from the internet. You can orally discuss solutions with other students, but any collaboration that involves writing things down will be considered as cheating and earn a zero credit. If you are stuck, please talk to the instructor. Python does not limit the size of the integers so you can run it on arbitrarily large integers (bignums). Write a verifier function that multiplies the given factors together and checks if it matches the input number, i.e., verify(12, factors(12)) should return true. Test your program thoroughly by running it on large integers.

2. In the report, include the code and a derivation of the running time of your algorithm (a) assuming that multiplications (and additions) take constant time and (b) assuming that multiplication and division of n-bit numbers take O(n^2) time and additions and subtractions take O(n) time.

3. The size of the input n is usually measured by the number of bits needed to represent the input. But here we can use decimal digits since it is directly proportional to the bits. Give a table $T(n)$ vs. n from your experimental results. Does your table closely match one of the running time functions derived in 2? How large can n be so that $T(n)$ is approximately 5 minutes. What if $T(n)$ is 5 hours? 5 days? Factoring is a fundamental crypto-primitive that underlies modern cryptography. What size of n makes it practically impossible to factorize, e.g., $T(n) > 10$ years.

How to submit?

- Label the source file 'hw1.py' and the report report1.pdf.
- Upload report1.pdf on the Canvas site before the deadline.
- Upload the source hw1.py also before the deadline at COE's teach interface for Homework1 at https://teach.engr.oregonstate.edu/ Links to an external site.
- Please test your program thoroughly before you submit.

Rubric

- Program runs correctly: 5 points.
- Complexity analysis: 3 points
- Table of results and estimates: 2 points