python assignment

1. Write a program that prompts the user to enter three integers. The program then reports the values of the three integers, ordered from least to greatest. In the case of repeated values, report the same value multiple times. (Use conditionals to determine the order of the integers. Avoid using built-in sorting features of the language, as the purpose of this problem is to get the conditionals correct.)

Sample input:

Please enter the first integer: 1

Please enter the second integer: 5

Please enter the third integer: 3

From least to greatest, the order is: 1, 3, 5

2. Write a program that prompts the user to enter three floats a, b and c, representing the coefficients of the polynomial $ax^2 + bx + c$. Note that all three values are allowed to be 0.0. The program then reports the roots of the polynomial and their multiplicities (note that the roots may be complex if a ≠= 0.0 or if all three coefficients are 0.0). If the polynomial is the zero polynomial, report that the polynomial is the zero polynomial and that all complex numbers are roots (you may ignore multiplicities in this instance). When the roots are real, report them using float values (not complex values).

Sample input:

Please enter coefficient a: 2

Please enter coefficient b: 6

Please enter coefficient c: 4

The two distinct roots of the polynomial are -2.0 and -1.0, each of multiplicity 1.

Sample input:

Please enter coefficient a: 1

Please enter coefficient b: 2

Please enter coefficient c: 1

The only root of the polynomial is -1.0, of multiplicity 2.

Sample input:

Please enter coefficient a: 1

Please enter coefficient b: 4

Please enter coefficient c: 13

The two distinct roots of the polynomial are (-2-3j) and (-2+3j), each of multiplicity 1.

Sample input:

Please enter coefficient a: 0

Please enter coefficient b: 1.2

Please enter coefficient c: 4.6

The only root of the polynomial is -3.833333333333333, of multiplicity 1.

Sample input:

Please enter coefficient a: 0

Please enter coefficient b: 0

Please enter coefficient c: 4

The polynomial is a non-zero constant, and thus has no roots!

Sample input:

Please enter coefficient a: 0

Please enter coefficient b: 0

Please enter coefficient c: 0

The polynomial is the zero polynomial, and thus every complex number is a root!

3. Write a program that prompts the user to enter a positive integer n. The program should then compute and report the sum of the fourth powers of the integers from 1 to n inclusive using a for loop. The program should also compute the sums of the fourth powers of the integers using the formula:

$$\sum_{k=1}^{n} k^4 = \frac{(n)(n+1)(2n+1)(3n^2+3n-1)}{30},$$

and report this value to the user as well.

Sample input:

Please enter a positive integer n: 10

The total computed using a for loop is: 25333

The total computed using the formula is: 25333

Sample input:

Please enter a positive integer n: 500

The total computed using a for loop is: 6281291666650

The total computed using the formula is: 6281291666650

Sample input:

Please enter a positive integer n: 3000000

The total computed using a for loop is: 486000405000089999999999999900000

The total computed using the formula is: 486000405000089999999999999900000

4. Revise program from to account for some additional details:

(a) Use correct grammar for when the user enters one of an item (e.g. do not have your program report “1 apples” but instead “1 apple” in this case).

(b) If the user enters a negative value for the number of an item, prompt them to enter the value again until they enter a non-negative number of items. (Requires iteration. You do not need to account for the possibility of the user entering something that cannot be interpreted as an integer.)

```
# This program is intended to output a bill for a fruit stand

num_apples = int(input("Please enter the number of apples: "))
num_bananas = int(input("Please enter the number of bunches of bananas: "))
num_oranges = int(input("Please enter the number of oranges: "))
num_lemons = int(input("Please enter the number of lemons: "))

num_total = num_apples + num_bananas + num_oranges + num_lemons

price_apples = 0.50
price_bananas = 1.75
price_oranges = 0.75
price_lemons = 1.50

tot_apples = num_apples*price_apples
tot_bananas = num_bananas*price_bananas
tot_oranges = num_oranges*price_oranges
tot_lemons = num_lemons*price_lemons

bill_tot = tot_apples + tot_bananas + tot_oranges + tot_lemons

print("")

print(f"The total bill for {num_apples} apples is: ${tot_apples:.2f}")
print(f"The total bill for {num_bananas} bunches of bananas is:
${tot_bananas:.2f}")
print(f"The total bill for {num_oranges} oranges is: ${tot_oranges:.2f}")
print(f"The total bill for {num_lemons} lemons is: ${tot_lemons:.2f}")

print("")

print(f"The total number of items purchased is: {num_total}")
print(f"The overall total bill is: ${bill_tot:.2f}")
```

5. Write a program that prompts the user to enter a positive integer n. The program then computes the sum of all the integers from 1 to n inclusive that are divisible by either 3 or by 5, but not by both. Report the sum of the appropriate integers to the user. Use a for loop and conditionals to complete this problem, even if you can think of a more efficient way that uses neither.

Sample input:
Please enter a positive integer n: 10
The sum of the integers from 1 to 10 satisfying the conditions is 33.

Sample input:
Please enter a positive integer n: 20
The sum of the integers from 1 to 20 satisfying the conditions is 83.

Sample input:
Please enter a positive integer n: 1000
The sum of the integers from 1 to 1000 satisfying the conditions is 201003.

Sample input:
Please enter a positive integer n: 2000000
The sum of the integers from 1 to 2000000 satisfying the conditions is 800000000003.

6. Write a program to approximate the sum of the following alternating series:

$$\sum_{k=0}^{\infty}(-1)^k \frac{k+1}{k^3+3k+1}$$

so that the error in the sum is smaller than a given tolerance. Your program should request the user to enter a positive error tolerance, and then repeatedly add terms in sequence until the error between the sum of the series and the partial sum is guaranteed to be strictly less than the tolerance (assuming the arithmetic were exact). Use the error bound from the alternating series test (described below) to determine when to stop adding terms. Report the partial sum of the series and the number of terms included in the sum to the user. Note that you do not need to know (and should not compute) the actual sum of the series!

One form of the error bound in the alternating series test is as follows:

Let a_k be a sequence of non-negative values satisfying $|a_{k+1}| \leq |a_k|$ for all $k \geq 0$, with $\lim_{k\to\infty} a_k = 0$. Then the series $S = \sum_{k=0}^{\infty}(-1)^k a_k$ converges, and furthermore, with $S_n = \sum_{k=0}^{n}(-1)^k a_k$, we have that

$$|S - S_n| \leq |a_{n+1}|$$

(i.e. the error in approximating the series by the partial sum up to and including the term of index n is less than or equal to the magnitude of the first omitted term).

Sample input:
Please enter a positive error tolerance: 10
The total computed is: 0.0.
The number of terms included in the sum is: 0.

Sample input:
Please enter a positive error tolerance: 1
The total computed is: 1.0.
The number of terms included in the sum is: 1.

Sample input:
Please enter a positive error tolerance: .1
The total computed is: 0.6918918918918919.
The number of terms included in the sum is: 4.

Sample input:
Please enter a positive error tolerance: .01
The total computed is: 0.7363621812194436.
The number of terms included in the sum is: 11.

Sample input:
Please enter a positive error tolerance: .001
The total computed is: 0.7320499100069268.
The number of terms included in the sum is: 33.

Sample input:
Please enter a positive error tolerance: 1e-8
The total computed is: 0.7315637257173737.
The number of terms included in the sum is: 10001.