Week 5 Assignment

Questions

Question 1

Write a program that reads integers from the user and stores them in a list. Your program should continue reading values until the user enters 0. Then it should display all of the values entered by the user (except for the 0) in ascending order, with one value appearing on each line. Use either the `sort()` method or the `sorted()` function to sort the list.

Question 2

When analyzing data collected as part of a science experiment it may be desirable to remove the most extreme values before performing other calculations. Write a function that takes a list of values and an non-negative integer, n, as its parameters. The function should create a new copy of the list with the n largest elements and the n smallest elements removed. Then it should return the new copy of the list as the function's only result. The order of the elements in the returned list does not have to match the order of the elements in the original list.

Write a main program that demonstrates your function. It should read a list of numbers from the user and remove the two largest and two smallest values from it by calling the function described previously. Display the list with the outliers removed, followed by the original list. Your program should generate an appropriate error message if the user enters less than 4 values

Question 3

A proper divisor of a positive integer, n, is a positive integer less than n which divides evenly into n. Write a function that computes all of the proper divisors of a positive integer. The integer will be passed to the function as its only parameter. The function will return a list containing all of the proper divisors as its only result. Complete this exercise by writing a main program that demonstrates the function by reading a value from the user and displaying the list of its proper divisors. Ensure that your main program only runs when your solution has not been imported into another file.

Question 4

An integer, n, is said to be perfect when the sum of all of the proper divisors of n is equal to n. For example, 28 is a perfect number because its proper divisors are 1, 2, 4, 7 and 14, and 1 + 2 + 4 + 7 + 14 = 28.

Write a function that determines whether or not a positive integer is perfect. Your function will take one parameter. If that parameter is a perfect number then your function will return True. Otherwise it will return False. In addition, write a main program that uses your function to identify and display all of the perfect numbers between 1 and 10,000. Import your solution to previous question when completing this task.

Question 5

Create a program that reads integers from the user until a blank line is entered. Once all of the integers have been read your program should display all of the negative numbers, followed by all of the zeros, followed by all of the positive numbers. Within each group the numbers should be displayed in the same order that they were entered by the user. For example, if the user enters the values 3, -4, 1, 0, -1, 0, and -2 then your program should output the values -4, -1, -2, 0, 0, 3, and 1. Your program should display each value on its own line.

Question 6

In order to win the top prize in a particular lottery, one must match all 6 numbers on his or her ticket to the 6 numbers between 1 and 49 that are drawn by the lottery organizer. Write a program that generates a random selection of 6 numbers for a lottery ticket. Ensure that the 6 numbers selected do not contain any duplicates. Display the numbers in ascending order.

Bonus Questions

Bonus Question 1

A standard deck of playing cards contains 52 cards. Each card has one of four suits along with a value. The suits are normally spades, hearts, diamonds and clubs while the values are 2 through 10, Jack, Queen, King and Ace.

Each playing card can be represented using two characters. The first character is the value of the card, with the values 2 through 9 being represented directly. The characters “T”, “J”, “Q”, “K” and “A” are used to represent the values 10, Jack, Queen, King and Ace respectively. The second character is used to represent the suit of the card. It is normally a lowercase letter: “s” for spades, “h” for hearts, “d” for diamonds and “c” for clubs. The following table provides several examples of cards and their two-character representations.

Card	Abbr
Jack of Spades	Js
Two of Clubs	2c
Ten of Diamonds	Td
Ace of Hearts	Ah
Nine of Spades	9s

Begin by writing a function named createDeck. It will use loops to create a complete deck of cards by storing the two-character abbreviations for all 52 cards into a list. Return the list of cards as the function’s only result. Your function will not require any parameters.

Write a second function named shuffle that randomizes the order of the cards in a list. One technique that can be used to shuffle the cards is to visit each element in the list and swap it with another random element in the list. You must write your own loop for shuffling the cards. You cannot make use of Python’s built-in shuffle function.

Use both of the functions described in the previous paragraphs to create a main program that displays a deck of cards before and after it has been shuffled. Ensure that your main program only runs when your functions have not been imported into another file.

> A good shuffling algorithm is unbiased, meaning that every different arrange- ment of the elements is equally probable when the algorithm completes. While the approach described earlier in this problem suggested visiting each element in sequence and swapping it with an element at a random index, such an algo- rithm is biased. In particular, elements that appear later in the original list are more likely to end up later in the shuffled list. Counterintuitively, an unbiased shuffle can be achieved by visiting each element in sequence and swapping it to a random index between the position of the current element and the end of the list instead of randomly selecting any index.

Bonus Question 2

Tokenizing is the process of converting a string into a list of substrings, known as tokens. In many circumstances, a list of tokens is far easier to work with than the original string because the original string may have irregular spacing. In some cases substantial work is also required to determine where one token ends and the next one begins.

In a mathematical expression, tokens are items such as operators, numbers and parentheses. The operator symbols that we will consider in this (and subsequent) problems are *, /, ^, - and +. Operators and parentheses are easy to identify because the token is always a single character, and the character is never part of another token. Numbers are slightly more challenging to identify because the token may consist of multiple characters. Any sequence of consecutive digits should be treated as one number token.

Write a function that takes a string containing a mathematical expression as its only parameter and breaks it into a list of tokens. Each token should be a parenthesis, an operator, or a number. (For simplicity we will only work with integers in this problem). Return the list of tokens as the function’s only result.

You may assume that the string passed to your function always contains a valid mathematical expression consisting of parentheses, operators and integers. However, your function must handle variable amounts of whitespace (including no whitespace) between these elements. Include a main program that demonstrates your tokenizing function by reading an expression from the user and printing the list of tokens. Ensure that the main program will not run when the file containing your solution is imported into another program.

Bonus Question 3

Some mathematical operators are unary while others are binary. Unary operators act on one value while binary operators act on two. For example, in the expression 2 * -3 the * is a binary operator because it acts on both the 2 and the –3 while the - is a unary operator because it only acts on the 3.

An operator's symbol is not always sufficient to determine whether it is unary or binary. For example, while the - operator was unary in the previous expression, the same character is used to represent the binary - operator in an expression such as 2 - 3. This ambiguity, which is also present for the + operator, must be removed before other interesting operations can be performed on a list of tokens representing a mathematical expression.

Create a function that identifies unary + and - operators in a list of tokens and replaces them with u+ and u- respectively. Your function will take a list of tokens for a mathematical expression as its only parameter and return a new list of tokens where the unary + and - operators have been substituted as its only result. A + or - operator is unary if it is the first token in the list, or if the token that imme- diately precedes it is an operator or open parenthesis. Otherwise the operator is binary.

Write a main program that demonstrates that your function works correctly by reading, tokenizing, and marking the unary operators in an expression entered by the user. Your main program should not execute when your function is imported into another program.

Bonus Question 4

The Sieve of Eratosthenes is a technique that was developed more than 2,000 years ago to easily find all of the prime numbers between 2 and some limit, say 100. A description of the algorithm follows:

```
Write down all of the numbers from 0 to the limit
Cross out 0 and 1 because they are not prime

Set p equal to 2
While p is less than the limit do
    Cross out all multiples of p (but not p itself)
    Set p equal to the next number in the list that is not crossed out
Report all of the numbers that have not been crossed out as prime
```

The key to this algorithm is that it is relatively easy to cross out every nth number on a piece of paper. This is also an easy task for a computer—a for loop can simulate this behavior when a third parameter is provided to the range function. When a number is crossed out, we know that it is no longer prime, but it still occupies space on the piece of paper, and must still be considered when computing later prime numbers. As a result, you should not simulate crossing out a number by removing it from the list. Instead, you should simulate crossing out a number by replacing it with 0. Then, once the algorithm completes, all of the non-zero values in the list are prime.

Create a Python program that uses this algorithm to display all of the prime numbers between 2 and a limit entered by the user. If you implement the algorithm correctly you should be able to display all of the prime numbers less than 1,000,000 in a few seconds.

> This algorithm for finding prime numbers is not Eratosthenes' only claim to fame. His other noteworthy accomplishments include calculating the circum- ference of the Earth and the tilt of the Earth's axis. He also served as the Chief Librarian at the Library of Alexandria.