1. Consider a list of positive numbers (L) and a target key (t). Find a contiguous sub-list within L of size >=1 such that the sum of the sublist = t. Not all the elements in the sublist will be contiguous and not separated.
    a. Case 1: Find the first sublist and return it else -1
    b. Case 2: Find the smallest sublist in terms of length of sublist and return it, else -1.
    c. Now consider that the list can contain positive and negative numbers, and find all sublists that sum to zero.

2. Consider two lists, L1 and L2. Write a program to check if L2 is a contiguous sublist of L1.

3. Write a program that takes a list as input and a rotation index and performance cyclic rotation of the list along that index.
    a. In a clockwise manner,
    b. In an anti-clockwise manner

4. Write a that takes a list of lists as input, a 2-D list with one-row input together comma separated on each line.
    a. Print the transpose of the 2-D list.
    b. Print the multiplication of A^TA
    c. Print the multiplication of AA^T
    d. Check if b=c?

5. Only single pass i,e iteration over the list of known size N is allowed.
    a. Given a list of numbers: Swap the first element with the last element, a second element with the second last element, and so on.
    b. Given a list consisting of only zeros and one, generate a list where zeros and ones are segregated i,e, all zeros on one side and all 1s on other side.
    c. Given a list of numbers and a step value/common difference d, check if the given list is an arithmetic series or not.
    d. Give a list of numbers and power value/common ratio r, and check whether the given list is a geometric series.
    e. Without any sorting: You are allowed to employ other data structures but not sort any DS.
        i. Find the minimum and maximum elements in the list.
        ii. Find the second largest and second smallest element in the list.
        iii. Find the k smallest and largest element in the list, where k is given by the user. Is this even possible without sorting?
    f. Write a program to find whether an element is present in the list or not. If not, return the nearest smallest and nearest biggest element to it. For example, for [4,1,2] and searching for 3, we return 2,4; the smallest number of 3 is closest to 3, and similarly for 4. If 3 is present in the array, we return 3,3.

    Note in the Q3 using list functions like sum(), sort(), and len() counts as pass as that also requires iterating over the whole list.

6. Fun with Prime numbers:
    a. You are given a list of numbers from 1 to N (both inclusive) as input. Write a program to employ the Sieve of Eratosthenes to return the sum of all prime numbers unto N.
    b. You are given a number P and determine if the number is semi-prime. A semi-prime is a number that is not prime itself but can be expressed as a product of two prime numbers.
    c. We try to determine the number of possible combinations representing a number N via Levy's conjecture, which states that an odd number 2N + 1 can be expressed as a sum p+2q where p and q are both prime. For example, for N = 9, the odd number is 19, and 19=5+2*7 or 19=13+2*3. Write a program that takes a number N <=20 and determines the number of ways the odd number 2N+1 can be expressed. Reference solution https://oeis.org/A046927/list.

7. L-Norm distance measure. You are given two 1-D arrays, each of the same length n, and a parameter p your aim is to print the L-Norm distance given by the formula:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Reference: https://medium.com/@kunal_gohrani/different-types-of-distance-metrics-used-in-machine-learning-e9928c5e26c7

8. Write a program maintaining a count of alphabets entered so far, until -1 is encountered. Once -1 is encountered, print the frequency histogram using "*" in alphabetical order in
   a. Horizontal manner
   b. Vertical manner

   For example, when the input is A,B,Z,C,A,A,Z,Z,Z,B,B,Z,Z,-1, then the frequency of A is 3, B is 3, Z is 6, and C is 1.

   Horizontal:

```
A * * *
B * * *
C *
Z * * * * * *
```

   Vertical:

```
         *
         *
         *
*  *     *
*  *     *
*  *  *  *
A  B  C  Z
```

9. Similar to the above input, but this time, we are not interested in counting but rather capturing the unique occurrences of each incoming alphabet. What data structure should we use to print the unique alphabets if:
   a. Order in which they entered the system should be maintained.
   b. Order does not matter.

10. Mimicking list functionality:
    a. Write a program that accepts a list and a number d to be deleted from the list and returns the list with the deleted element if d is present, -1 otherwise. You are not allowed to use the predefined remove or del operations.
    b. Same as above, but you are given the index to remove instead of the value itself.
    c. Write a program that takes start, end and jump values as input and mimics the slice operation in a list. I.e., implement slice functions on your own. Your slicing operation should take care of -ve indexing, cases where only start or end is given, handle index out-of-bound operations, and so on. Compare your results against the slicing operation [start:end: jump] to validate results. Try to cover as many cases as possible. Employ your knowledge to handle any errors and exceptions gracefully.

11. Mimicking dictionary functionality:
    a. Write your own function to perform sorting on a dictionary with numeric keys and values based on keys.
    b. Same as above, but sorting based on values.
    c. Sort first on key, and multiple keys have the same rank, then sort on respective values.
    d. Same as above, but sort on value, and if multiple values have rank, then sort on respective keys.

12. You are given two dictionaries. Your task is to generate two dictionaries, U and I, which represent the union and intersection of the two dictionaries based on keys. The values should be updated according to the following setup:
    a. Assuming values are all numeric in both dictionaries, the updated values will be the sum of the two.
    b. Assuming values can be of different datatypes, each key will now have a list of values to store values coming from each dictionary. Ensure the data types of values for the final U and I are the same, i.e. list.