1. In this question, we will test if a generated password in some website is valid or not. A user will give a username and a password as string inputs. Our task is to verify if the password is acceptable or not, which is checked from the following rules:

a. The password can not be the same as the username, and it can't contain the username as a part of it

b. There should be at least 8 characters in the password

c. The maximum length of the password can be 15

d. It should contain at least one lowercase, one uppercase, and at least two characters from the following: $, *, +, /, #, @, <, >, ?, _, !

e. There can be maximum 5 consecutive digits in the password

f. There can't be more than 5 repeated characters

You are not allowed to use/call any String.h library functions. Feel free to write your own functions and invoke them.

Example 1:

username: harrisjh

password: h67dQ*+R62

Acceptable

Example 2:

username: david1

password: 45Cxdavid1*?I

Not acceptable

Example 3:

username: jdoe123

password: uPG10/rrX

Not acceptable

2. A genetic sequence is a string formed from a four-letter alphabet {Adenine (A), Thymine (T), Guanine (G), Cytosine (C)} of biological macromolecules referred to together as the DNA bases. A gene is a genetic sequence that contains the information needed to construct a protein. Following are some examples of parts of gene structures:

ATTGCATGGACCTGCGAATCTGAGGCTA

GGTCCAAGAGATTAACTGTGCAAAC

CTAGGCTGCAAGTCACAATCGTGTGTAACAAGGT

Dr. Xu is performing some study with the genes of Sars Covid19 virus and some of its variants. In the study, he needs to find the similarity between two genes coming from two different Covid variants by means of a "similarity score". The goal is that, if the two genetic sequences are similar enough, we might expect them to have similar functions. Dr. Xu wants to measure the similarity of two genetic sequences by the following procedure. The two sequences will be aligned, and we will pay a penalty for each mismatch between two corresponding characters. However if the two sequences do not have the same length, then "gaps" (represented by '-') will be inserted to the shorter sequence to make the two sequences have the same length. For simplicity, we consider maximum 2 gaps in the sequences (that means, the length of the two sequences will differ by maximum 2 characters). There can be many possibilities of inserting the gap(s) in a sequence. Our goal is to find the gap positions for which the "similarity score" is the minimum. The following rules are followed for the penalty and cost:

Penalty	Cost
Per gap	2
Per mismatch	1
Per match	0

For example, consider the following two sequences: AACAGTTACC and TAAGGTCA. Two possible alignments can be:

Example 1

Sequence1 AACAGTTACC

Sequence2 TAAGGTCA - -

Penalty 10 1 1 0 0 1 0 2 2

Similarity score = 8

Example 2

Sequence1 AACAGTTACC

Sequence2 TA - AGGT- CA

Penalty 10 2 0 0 10 2 01

Similarity score = 7

There can be many other combinations of inserting gaps. Your task is to write a C program which takes the two sequences as two strings from the user as input.

If the sequences have different lengths, then find the alignment of the two sequences for which the Similarity score is minimum.

If the sequences have same length, then simply find the similarity score by the above mentioned rules. Your output should display the aligned sequences (along with the gaps represented by '-'), as well as the similarity score.

You are not allowed to use/call any String.h library functions. Feel free to write your own functions and invoke them. (May use ASCII code etc to identify the special characters)

3. Given three lines of the form ax+by=c, your program has to compute their points of intersection and hence the area of the triangle described by them. Take the co-efficients (a, b, c) of each line in main(). Then, write a function that takes as input (as parameter) the coefficients (a,b,c) of two lines and computes their intersection point if they are not mutually parallel. Call this function three times from main() and print required results (intersection points, side length, area, messages etc) in main() as shown in examples. You may use an array to pass the intersection points from function to main().

You can use sqrt function of math.h.

Examples:

$ Enter the coefficients (a, b, c) of three lines:
Line 1: 1 0 0
Line 2: 0 1 0

Line 3: 1 1 1

Point of intersection between L1 and L2 = (0.000000, -0.000000).
Point of intersection between L2 and L3 = (1.000000, 0.000000).
Point of intersection between L3 and L1 = (-0.000000, 1.000000).
Side lengths = 1.000000, 1.414214, 1.000000.
Area = 0.500000

$ Enter the coefficients (a, b, c) of three lines:
Line 1: 1 0 0
Line 2: 1 0 1
Line 3: 1 1 1
Lines 1 and 2 are parallel; so no point of intersection.

4. User supplies a list of n distinct integers sorted in increasing order. First check if the list is sorted (in increasing order). He/She also supplies a key x, which is an integer. Assume that n lies in [1,20].
Store the list as a 1D array and write a program to search x in the array using (a) linear search and (b) binary search. Implement the linear search in **linear_search()** function and binary search in **bin_search()** function. Both the functions should take the list and the key x as argument and should return the index (for successful search, -1 for unsuccessful search).

Ask the user to choose the required searching algorithm and invoke the respective function.
In case of successful search, print in the main() the position of x in the array. Position of an element is indicated by its array index. In case of unsuccessful search, print -1.
Implement the binary search using iterative procedure, NOT by recursion.
In both the cases, print the messages in main function.

Example 1:

Enter the number of elements n: 5

Enter the elements in sorted order: 4, 8, 12, 18, 21

Enter the key element x, which you want to search: 12

Choose the algorithm: Type 1 for linear search and 2 for binary search

2

Executing the binary search algorithm

Successful search

Element 12 has been found at the index 2

Example 2:

Enter the number of elements n: 5

Enter the elements in sorted order: 8, 4, 12, 18, 21

Not sorted. Enter again

Enter the elements in sorted order: 4, 9, 12, 18, 21

…….

……