LAB-PRAC-08_ARR-STR

Il fratello di Fibonacci (p1v1d1)

Il fratello di Fibonacci [10 marks]

Problem Statement
We have seen the Fibonacci sequence, and its generalization, the the Lucas sequence, in previous weeks. Today, let us look at another generalization of the Fibonancci sequence where, instead of the first two non-negative integers deciding the sequence, the first k non-negative integers decide the sequence. Let us call this the k-FF-sequence (short for kth order fratello di Fibonacci).

The n-th number in the k-FF sequence is defined to be the sum of the previous k numbers in the k-FF sequence. The first k numbers in the k-FF sequence are 0, 1, ..., k-1. We will give you two strictly positive integers k and n, separated by a single space, as input. As output, you have to output the (k+1)-th number, as well as the n-th number in the k-FF sequence, on the same line but separated by a space.

We promise that k and n will both be less than or equal to 49 and your outputs will always fit inside an int variable.

P.S.: the title of this problem means "Fibonacci's brother" in French.

Caution: Be careful about extra/missing lines and extra/missing spaces in your output.

HINT: Use an array to store the first k k-FF numbers and also calculate the subsequent ones. Remember, we will never give k,n > 100.

EXAMPLE 1:
INPUT
2 5

OUTPUT:
1 3

Explanation: the first two non-negative integers are 0 and 1. So the 2-FF-sequence will be as follows
2-FF1: 0
2-FF2: 1
2-FF3: 0 + 1 = 1
2-FF4: 1 + 1 = 2
2-FF5: 1 + 2 = 3
so the (k+1)-th 2-FF number is the 3rd 2-FF number which is 1 and the 5th 2-FF number is 3.

EXAMPLE 2:
INPUT
3 6

OUTPUT:
3 11

Explanation: the first three non-negative integers are 0, 1, and 2. So the 3-FF-sequence will be as follows
3-FF1: 0
3-FF2: 1
3-FF3: 2
3-FF4: 0 + 1 + 2 = 3
3-FF5: 1 + 2 + 3 = 6
3-FF6: 2 + 3 + 6 = 11
so the (k+1)-th FF number is the 4th 3-FF number which is 3 and the 6th 3-FF number is 11.

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
2 13	1 144
3 10	3 125
5 10	10 149
6 20	15 109065
50 20	1225 19
4 20	6 116072

Hidden Palindrome (p1v2d1)

Hidden Palindrome [20 marks]

Problem Statement
The input will contain a string containing n characters (we promise that n will be equal to or less than 49). Read this string into a character array (lets call it str) (i.e. 1st character of the string is stored in str[0], 2nd character in str[1], and last character in str[n-1]. Additionally, s[n] = â€˜\0â€™ should be ensured by Mr C if you read the string from the input properly).

Find the length of the largest substring of str (including str itself) which is a palindrome. Print this length as the first line of your output. In the second and third lines of your output, print the indices of the starting character and the ending character of the this largest substring.

If there are two or more such substrings which are palindromes and of largest length, print the starting and ending indices of the substring with the least starting index.

We promise we will not give you a string that contains whitespace or non-printable characters like space/tab/newline. We also promise we will never give you an empty string as input.

Caution

1. A single letter/character is trivially a palindrome too, although a very short one.
2. Be careful about extra/missing lines and extra/missing spaces in your output.

HINT: The strlen() function can help you find the length of a string. Given a character array, it returns the number of characters in the array uptil, but not including, the first occurrence of the null character in the array. Include the header file string.h to use this function.

EXAMPLE 1 :
INPUT
cbcdc

OUTPUT:
3
0
2

Explanation: â€œcbcâ€□ and â€œcdcâ€□ both are substrings in the given string which are palindromes. Since there are no substrings of size 4 or larger which are palindromes, the output is 3. The substring with the smallest starting index is â€œcbcâ€□, with starting index 0 and ending index 2.

EXAMPLE 2 :
INPUT
abcde

OUTPUT:
1
0
0

Explanation: The only substrings of the given string which are palindromes are of size 1 (i.e. the trivial palindromes â€œaâ€□,â€□bâ€□,â€□câ€□,â€□dâ€□,â€□eâ€□). Hence, the length of the largest palindrome is 1. The substring with the smallest starting index is â€œaâ€□, with starting index 0 and ending index 0.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are three lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 50% weightage and the next two lines carry 25% weightage each. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
abcba	5 0 4
abcbd	3 1 3
baaaaaaaaaaaaaaaaaaa	19 1 19
s	1 0 0
abcdcdcdefefefegggg	7 8 14

wowowmomom	5 0 4

Hush Hush Hash (p1v3d1)

Hush Hush Hash [20 marks]

Problem Statement
The concept of a hashing is critical in computer science. Hashing is the art and science of converting a given piece of data, like a string, into a single number which can be stored. For instance, the CC (or even GMail, Piazza, Gradescope etc) does not store your password anywhere. It just hashes your password stores the hash. When you try to logon to CC Email and enter your password, the CC hashes what you have entered and compares this hash value and compares it to the hash value that it has stored. If the two match, you are given login otherwise, you are denied login.

The CC etc use sophisticated and cryptographically secure hash functions like SHA etc. In this question, we will design two very simple hash functions. Your input will be a string with n characters where n will be equal to 499 or less. We promise that the characters will either be English letters 'A', 'x', 'E' etc or else digits '0', '6' etc. You have to take this string and convert this into a number as follows:

Assign every character a number in the following manner

1. Digits get assigned their own value i.e. '0' gets assigned value 0, '6' gets assigned value 6 etc.
2. Letters get assigned a value equal to their position in their alphabet. E.g. 'A' and 'a' both have value 1, 'C' and 'c' both have value 3, 'Z' and 'z' both have value 26 etc.
3. Note that the values as calculated above are always non-negative integers.

Suppose your string is sitting inside a character array called str with the n characters stored in str[0] to str[n-1] and str[n] being the NULL character '\0' which Mr C should himself put in if you read the string from the input properly. Calculate the following two hash values for this string. For a character x let val(x) denote its value as calculated above.

$$
\begin{align*}
hash1(str) &= \sum_{i=0}^{n-1} val(str[i])\\
hash2(str) &= \left(\prod_{i=0}^{n-1} val(str[i]) \right) \% 100000
\end{align*}
$$

Print the two hash values calculated above (they will always be non-negative integers), in two separate lines in your output.

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.
2. Once you have gone back home after finishing today's lab, you may want to look up the term "Hash function" on the internet (don't browse the internet during the lab though - it is banned)

HINTS:

1. The product of 499 numbers may be very large (and may overflow the limit of int/long) but the hash value in hash2 is always between 0 and 99999. Use the modulus trick we earlier saw that (a * b) % c = ((a % c) * (b % c)) %c
2. You may want to use the strlen function from string.h to find out the length of a string

----

EXAMPLE 1:
INPUT
1234

OUTPUT:
10
24

EXAMPLE 2:
INPUT
AbCd

OUTPUT:
10
24

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries 50% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
Zephyr	98 88000
123456789	45 62880
Iwonderhowmanyletterswouldittaketobreachtheintegerlimit987654321	718 0
88888888888888888888	160 46976
HelloESC101	81 0
18181818181818181818181818181818181818181818181818181818181818181818181818181818	360 44576

Maximum Match (p2v1d1)

Maximum Match [10 marks]

Problem Statement
In the first line of the input, you will be given two integers n and k. We assure you that n will be strictly positive but less than or equal to 1000. In the second line of the input, you will be given n integers sorted in non-decreasing order, with integers separated by a single space. You need to find out if there are any two distinct elements in the list such that the sum of the two elements is equal to k.

Your output should be in two lines.

1. Case 1: there are no two numbers in the list that sum to k. In this case, in the first line, print the words "No Match" (without quotes).
2. Case 2: there are exactly two numbers in the list of numbers that sum to k. In this case, in the first line, output the two numbers, which sum up to k. The two numbers should be printed in increasing order, separated by a single space. In the second line, print the locations at which these numbers occur in the list. The locations should be printed separated by a single space.
3. Case 3: there are more than one pair of numbers in the list that sum to k. In this case, choose the pair whose sum of locations is the maximum (an illustrative example is given below) and output the numbers and their locations as directed in Case 2 above.

Caution

1. When printing locations, use the human convention where the first location in the list is 1 (not the array index convention where locations begin with 0).
2. For the purposes of this question, two numbers in the list are distinct if they occur at distinct locations, even if they share the same value. Thus, the list 1 1 2 3 4 consists of two distinct elements of value 1.
3. Be careful about extra/missing lines and extra/missing spaces in your output.

HINT: You may want to use an array to store the list of numbers
----

EXAMPLE 1:
INPUT
8 11
1 2 3 8 8 8 10 11

OUTPUT:
3 8
3 6

Explanation: although the pairs (3,8) (note that 8 occurs thrice in the list) and (1,10) both add up to 11, the pair (3,8) occurs at the locations (3,6) in the list whereas (1,10) occurs at locations (1,7) since 3+6 > 1+7, we chose the pair (3,8).

EXAMPLE 2:
INPUT
6 10
1 2 3 4 5 21

OUTPUT:
No Match

Explanation: no two distinct numbers in the list add up to 10.

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
8 6 -5 0 3 3 4 4 5 6	0 6 2 8
6 10 -4 -2 4 8 14 16	-4 14 1 5
8 11 1 2 3 8 8 8 10 11	3 8 3 6
1 4 2	No Match
6 -2 -5 -4 -1 -1 10 12	-1 -1 3 4

El secreto de sus I (p2v2d1)

El secreto de sus I [20 marks]

Problem Statement
You will be given a string containing at least 1 and at most 999 characters. The string will only contain upper case English alphabets . In the first line, output the number of times the letter 'I' appears in the string. In the next line, you have to encrypt the sting in the following manner -- shift every letter in the English alphabet by the number of times the letter 'I' appeared.

For example, if the letter 'I' appears twice in the string, then we will English letters by 2, 'A' would become 'C', 'P' would become 'R', 'Z' would become 'B' etc. In the second line of your output, print the string from the first point the letter 'I' appeared in the string, but using the shifted alphabet (see an example below). If the letter 'I' does not appear in the original string at all, in the second line of the output, just print the original string again.

P.S. The name of this problem literally translates from Spanish to "The secret in their I" since in this problem the secret shift does lie in the letter I. The name is a play on the title of an Oscar-winning Argentinian crime drama called "El secreto de sus ojos" (The secret in their eyes).

Caution: Be careful about extra/missing lines and extra/missing spaces in your output.

HINTS:

1. You may want to use the strlen function by including string.h. The function tells you the length of a string (number of characters uptil but not including the first NULL character).
2. The strcpy function copies one string to another and may come in handy.
3. The strchr function in string.h may help you as well. It returns a pointer to the first occurrence of a given character in a string. If you feed that pointer into printf while using the %s format specifier, the string gets printed from that point onward. For example, printf("%s",strchr("Hello",'e')); will print "ello" without the quotes.

EXAMPLE 1:
INPUT:
NOCHANGE

OUTPUT:
0
NOCHANGE

EXAMPLE 2:
INPUT:
ABCDIABCDI

OUTPUT:
2
KCDEFK

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 25% weightage and the second line carries 75% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	
NOCHANGE	0 NOCHANGE
ABCDIABCDI	2 KCDEFK
IIIIIIIIIIIIII	16 YYYYYYYYY
ZZIIIIZIIIIZIIIIZIIIIZIIIIZIIIIZZ	24 GGGGXGGGG
ABCDEABCDEABCDEABCDEABCDEPQRSTPQRSTPQRSTPQRSTPQRSTPQRSTVWXYZVWXYZVWXYZVWXYZVWXYZVWXYZ	0 ABCDEABCDE
ABCDEABCDEABCDEABCDEABCDEPQRSTPQRSTPQRSTPQRSTPQRSTPQRSTVWXYZVWXYZVWXYZVWXYZVWXYZVWXYZIIII	4 MMMM

Star Replacement (p2v3d1)

Star Replacement [20 marks]

Problem Statement
You will given a string containing n characters where n is at least 1 but at most 499. We assure you that the string will contain only upper case English letters i.e. from 'A' to 'Z'. In the second line of the input, you will be given a single character which will also be an upper case English letter. You will have to change this string such that each occurrence of that character in the string is replaced by two consecutive asterisk/star characters i.e. "**" (without the quotes).

In the first line of your output, calculate and print the length of this new string. Next, print this new string in the second line of your output. We assure you that the new string will be of length 999 or less. If the character does not appear in the original string at all, the string will remain unchanged.

Caution

1. If you are storing the elements in a character array, then you will have to shift elements of the character array to the right since you will be replacing one character by two * characters. Use a loop carefully to shift elements to the right.
2. Be careful about extra/missing lines and extra/missing spaces in your output.

HINTS:

1. To do the above properly, you will find it helpful to first calculate the length of the final string. You anyway have to do this for the first part of the question.
2. The strlen function from string.h can be used to calculate the length of a string.

EXAMPLE:
INPUT
ABCDE
C

OUTPUT:
6
AB**DE

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 20% weightage and the second line carries 80% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

ABCDE C
ABCDE Z
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC C
ZZAAZZBBZZCCZZDDZZEEZZFFZZGGZZHHZZIIZZJJZZKKZZLLZZMMZZNNZZOOZZPPZZQQZZRRZZSSZZTTZZUUZZVVZZWWZZXXZZYYZZZZ Z
ZZAAZZBBZZCCZZDDZZEEZZFFZZGGZZHHZZIIZZJJZZKKZZLLZZMMZZNNZZOOZZPPZZQQZZRRZZSSZZTTZZUUZZVVZZWWZZXXZZYYZZZZ A
BCDEFGHIJKLMNOPQRSTUVWXYZAAAAAAAAAAA

A

Stronger together (p3v1d1)

Stronger Together [10 marks]

Problem Statement
In the first line of the input you will be given a strictly positive integer n. We assure you that n will be less than or equal to 1000. In the next line you will be given a list of n integers. Given this, there arise two cases

1. If there are no two distinct values in the list, print the word "DEFAULT" (without quotes).
2. If there are at least two distinct values in the list, then you have to find the two distinct values in the list that give the maximum possible sum. In the first line of the output, give the locations of these two numbers in the list in increasing order separated by a single space. In the second line of the output, give the elements at those locations, separated by a single space.
3. If there are more than one pair of elements in the array which give the same (maximum sum), then you have to give the pair and where the locations, as printed in the first line, are *lexicographically smallest*.

Given two pairs of integers (a,b) and (c,d), we find the lexicographically smaller one as follows

1. If a < c then (a,b) is lexicographically smaller. If a > c then (c,d) is lexicographically smaller
2. If a == c but b < d then (a,b) is lexicographically smaller. If a == c but b > d then (c,d) is lexicographically smaller
3. If a == c and b == d then obviously the two pairs are identical.

Caution

1. While printing the locations, use human convention for locations and not array index convention i.e. the first element of the array is to be considered to be at location 1, and not location 0.
2. The list need not be in sorted order.
3. For the sake of this question, two elements are considered distinct only if they do not have the same value. For example, if the list consists of 5 elements, 1 2 4 4 6, then there are only 4 distinct elements in the list, namely 1,2,4 and 6.
4. Be careful about extra/missing lines and extra/missing spaces in your output.

HINTS:

1. You may benefit from storing the list in an array first.

EXAMPLE 1:
INPUT
```
5
8 8 7 7 7
```

OUTPUT:
```
1 3
8 7
```

Explanation: Note that, the list has only two distinct elements (8 and 7), with 8 being at locations 1 & 2, while 7 is at indices 3,4,5. Clearly, the smallest pair, in terms of lexicographic ordering is (1, 3).

EXAMPLE 2:
INPUT
```
4
8 8 8 8
```

OUTPUT:
```
DEFAULT
```

Explanation: There are no two distinct elements in the list.

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
8 -5 0 6 3 4 4 5 6	3 7 6 5
8 5 5 5 -5 6 6 6 -2	1 5 5 6
1 8	DEFAULT
6 4 6 6 5 2 1	2 4 6 5
12 6 2 3 4 6 6 6 6 12 13 15 14	11 12 15 14
5 9 9 9 9 9	DEFAULT

Rigorous and repeated redaction (p3v2d1)

Rigorous and repeated redaction [20 marks]

Problem Statement
You will be given a non-empty string consisting only of upper case and lower case English alphabets in the first line of the input. The string will contain no more than 999 characters. The second line of the input will contain another non-empty string of only upper and lower case English alphabets with no more than 99 characters. The first string is a message your wish to publish online whereas the second string is sensitive material (e.g. passwords etc). You have to replace all instances of occurrences of the sensitive string in the original string with the letters "XXX" (without quotes). This process is often called *redaction*.

In the first line of your output, print how many times does the first letter of the message appear in the message either in lower or upper case. In the second line, print the redacted string. If the sensitive material is not present in the message at all, just print the original message in the second line.

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.
2. Although while counting how many times the first character of the message appears in the message, you have to count both upper and lower case occurrences, when checking for occurrences of the sensitive string, do so in a case sensitive manner (see example below).

HINTS:

1. The string.h library provides the strlen function to calculate the length of a string and the strstr function to find the occurrence of substrings in a given string. However, you need to be able to manipulate pointers to use strstr.
2. This question can be solved without using pointers as well.

EXAMPLE:
INPUT
hellohitHisismehighupinthesky
hi

OUTPUT:
6
helloXXXtHisismeXXXghupinthesky

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 25% weightage and the second line carries 75% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
hellohitHisismehighupinthesky hi	6 helloXXXtHisismeXXXghupinthesky
Thisisacleanmessagerequiringnoredaction secret	2 Thisisacleanmessagerequiringnoredaction
hellohitHisismehighupintheskyHihi hi	8 helloXXXtHisismeXXXghupintheskyHiXXX
wwwcseiitkacin www	3 XXXcseiitkacin

mypasswordishelloworldbutdonttellanyone helloworld	1 mypasswordisXXXbutdonttellanyone
IcanbeverycarelessandwriteanythingSECRETinthisstringanditwillallgetredactedbymynicecode secret	9 IcanbeverycarelessandwriteanythingSECRETinthisstringanditwillallge

strnrev (p3v3d1)

strnrev [20 marks]

Problem Statement
Some languages have offer library functions such as strrev to reverse a string (CLang does not though). We will implement a code in this question to reverse the first n characters of a given string - something that could be called str**n**rev. You will be given a non-empty string in the first line of the input and a non-negative integer n in the second line of the input. The string will contain no more than 999 characters and all those characters will either be lower or upper case English letters, digits, or spaces (no newlines though). Print the string with the first n characters reversed (see example below).

If n is 0, print the original string itself. If n is greater than the length of the string, print "ILLEGAL" (without quotes) and print nothing else.

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.

EXAMPLE:
INPUT
hello world
5

OUTPUT:
olleh world

Grading Scheme:
Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
hello world 5	olleh world
esc101 teaches C programming 100	ILLEGAL
esc101 teaches C programming 7	101cseteaches C programming
hello 5 5	olleh 5
it is a wonderful world 0	it is a wonderful world
i dream of jeanie 1	i dream of jeanie

Monster and Mini Multiply (p4v1d1)

Monster and Mini Multiply [10 marks]

Problem Statement
The input will be given in three lines, on the first two lines, you will be given two very large non-negative integers a and b with exactly 40 digits each. In the third line, you will be given a single non-negative, single digit integer x. You have to print the value of a*x + b as your output in a single line.

There should not be any leading zeros in your output. However, if the final answer is itself zero, just print a single "0" (without quotes) as your output.

Caution

1. The huge integers a and b will have exactly 40 digits in the way they are given to you. However, some of the leading digits in the representation of a and b may be zero (please see example below).
2. However, there should not be any leading zeros in your output.

3. Be careful about extra/missing lines and extra/missing spaces in your output. More specifically, do not print 0000000000000000000000000000000000000037 as your output. Print 37 instead.
4. Your output may have more than 40 digits

HINT: The integers you will be given cannot be stored even in a long variable. Consider storing them in an array.

EXAMPLE:
INPUT
9517330874833405504130385982694719818199
5691066055059061890234375756180617820379
4

OUTPUT:
43760389554392683906755919686959497093175

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
9517330874833405504130385982694719818199 5691066055059061890234375756180617820379 4	43760389554392683906755919686959497093175
0000000000000000000000000000000000000009 0000000000000000000000000000000000000001 4	37
0000000000000000000000000000000000000000 1234567890123456789012345678901234567890 4	1234567890123456789012345678901234567890
1234567890123456789012345678901234567890 0000000000000000000000000000000000000000 0	0
9999999999999999999999999999999999999999 9999999999999999999999999999999999999999 9	99999999999999999999999999999999999999990
0000875165097885844858410188187208148080 0000003083134303357327630204563117267823 7	6129238819504271336501521873574304383

Clash of the Substrings (p4v2d1)

Clash of the Substrings [20 marks]

Problem Statement
You will be given two strings str1 and str2 in two lines of the input. Both strings will contain less than or equal to 999 characters. You have to print the maximum length of a substring of str1 which is also a substring of str2. If there is no non-trivial substring of str1 which is also a substring of str2, print "0" (without quotes) as your output.

The name of this problem is a play on the movie "Clash of the Titans" which is based on Greek mythology.

Caution

1. There is only one line in your output.

HINTS:

1. The string.h library provides the strlen function to calculate the length of a string and the strstr function to find the occurrence of substrings in a given string. However, you need to be able to manipulate pointers to use strstr.
2. This question can be solved without using pointers as well.

EXAMPLE 1:
INPUT
ESC101 is a course on programming

Mr C teaches ESC101

OUTPUT:
6

Explanation: The largest substring common to both strings is "ESC101" (without quotes)

EXAMPLE 2:
INPUT
ESC101 is a course on programming
Mr C teaches ESC101 which is a course on programming

OUTPUT:
27

Explanation: The largest substring common to both strings is " is a course on programming" (without quotes)

Grading Scheme:
Total marks: **[20 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
ESC101 is a course on programming Mr C teaches ESC101	6
ESC101 is a course on programming Mr C teaches ESC101 which is a course on programming	27
abcdefgh ijklmnopq	0
Life Is Beautiful is a movie about the holocaust Yesterday, I watched a movie. It was called "Life Is Beautiful". I quite liked it.	10
abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvwxyz	26
Hotaru no Haka is a heartbreaking movie about children trapped in war Watashi wa eiga Hotaru no Haka ga suki	15

Personalizing Emails (p4v3d1)

Personalizing Emails [20 marks]

Problem Statement
I wish to write an email to the entire ESC101 class but I want to personalize it. In the first line of the input, you will be given a message string which will contain only printable characters and spaces/tabs (but no new lines obviously). In the second line of the input, you will be given the name of a student. The message string will contain no more than 999 characters and the name string will contain no more than 99 characters.

The message string may contain the substring "#NAME#" (without quotes). You have to replace all occurrences of this substring in the message string with the name string which you were given in the second line (see example below).

In the first line of the output, print the new message string. In the second line of the output, print how many times does the substring #NAME# appear in the message. If the substring #NAME# does not appear in the message string at all, just print the original message string in the first line and "0" (without quotes) in the second line.

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.

HINTS:

1. The string.h library provides the strlen function to calculate the length of a string and the strstr function to find the occurrence of substrings in a given string. However, you need to be able to manipulate pointers to use strstr.
2. This question can be solved without using pointers as well.

EXAMPLE 1:
INPUT
Hello #NAME#, Welcome to ESC101
Mr C

OUTPUT:
Hello Mr C, Welcome to ESC101

1

EXAMPLE 2:
INPUT
Hello John, Welcome to ESC101
Mr C

OUTPUT:
Hello John, Welcome to ESC101
0

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are two lines in your output. Printing each line correctly, in the correct order, carries some weightage. The first line carries 75% weightage and the second line carries 25% weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
Hello #NAME#, Welcome to ESC101 Mr C	Hello Mr C, Welcome to ESC101 1
Hello John, Welcome to ESC101 Mr C	Hello John, Welcome to ESC101 0
Hello #NAME#, Welcome to ESC101. Your registered name in the course is #NAME#. Please verify this Susan	Hello Susan, Welcome to ESC101. Your registered name in the course is Susan. Please verify this 2
Hello #NAME#, Welcome to ESC101. Your name in our records is #NAME#. So we will call you "#NAME#". Please verify this #NAME# and get back to us if there is a mistake. Tyler	Hello Tyler, Welcome to ESC101. Your name in our records is Tyler. So we will call you "Tyler". Please verify this Tyler and get back to us if there is a mistake. 4
Hello #NAME#, Welcome to ESC101. Your name in our records is #NA ME#. So we will call you "#NAME#". Please verify this #NAME and get back to us if there is a mistake. Carrie	Hello Carrie, Welcome to ESC101. Your name in our records is #NA ME#. So we will call you "Carrie". Please verify this #NAME and get back to us if there is a mistake. 2
Hello Amelia Amelia	Hello Amelia 0