

Lab Report on CSE 1200

Lab No. 2

**Submitted by Submitted to**

**Name**: Md. Shabir Khan Akash. Md. Asifur Rahman Sir

**Class**: 1st year, even semester. Lecturer

**Dept**.: Department of CSE Department of CSE, RUET

**Roll**: 1603108

**Section:** B

**Light Online Judge**

**Problem No. 1000**

**Greetings from LightOJ**

You are one of the most talented programmers and like to solve challenging problems. And my task is to make your life a bit complex by setting some easy looking hard brain storming problems such that you can sharpen your skills by practicing here. So, I wrote a code which shows a message like the following line:

**Greetings from LightOJ**

After that the code will select a random problem for you from my problems database based on your previously solved problems, your skills and your weaknesses. But while I was coding for implementing this great idea, I found that, all of my problems are scattered in **2** computers. So, I have to merge them before running my code.

Now you are given the number of problems in each of the computers, you have to find the total number of problems. You can safely assume that no problem is stored in multiple computers. So, all the problems are distinct.

**Input**

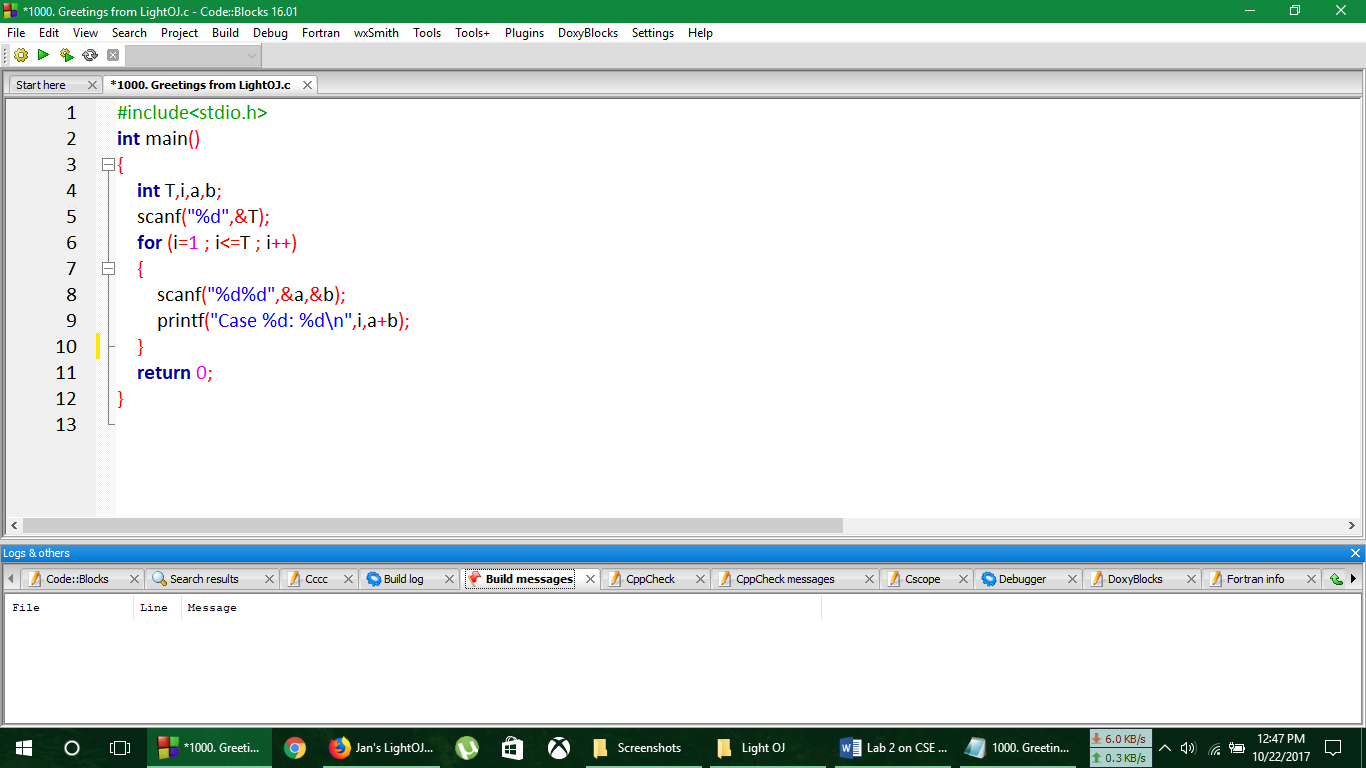
Input starts with an integer **T (≤ 125)**, denoting the number of test cases. Each case starts with a line containing two integers **a** and **b**. **a** denotes the number of problems in the first computer and **b** denotes the number of problems in the second computer. You can safely assume that **a** and **b** will be non-negative and not greater than **10**.

**Output**

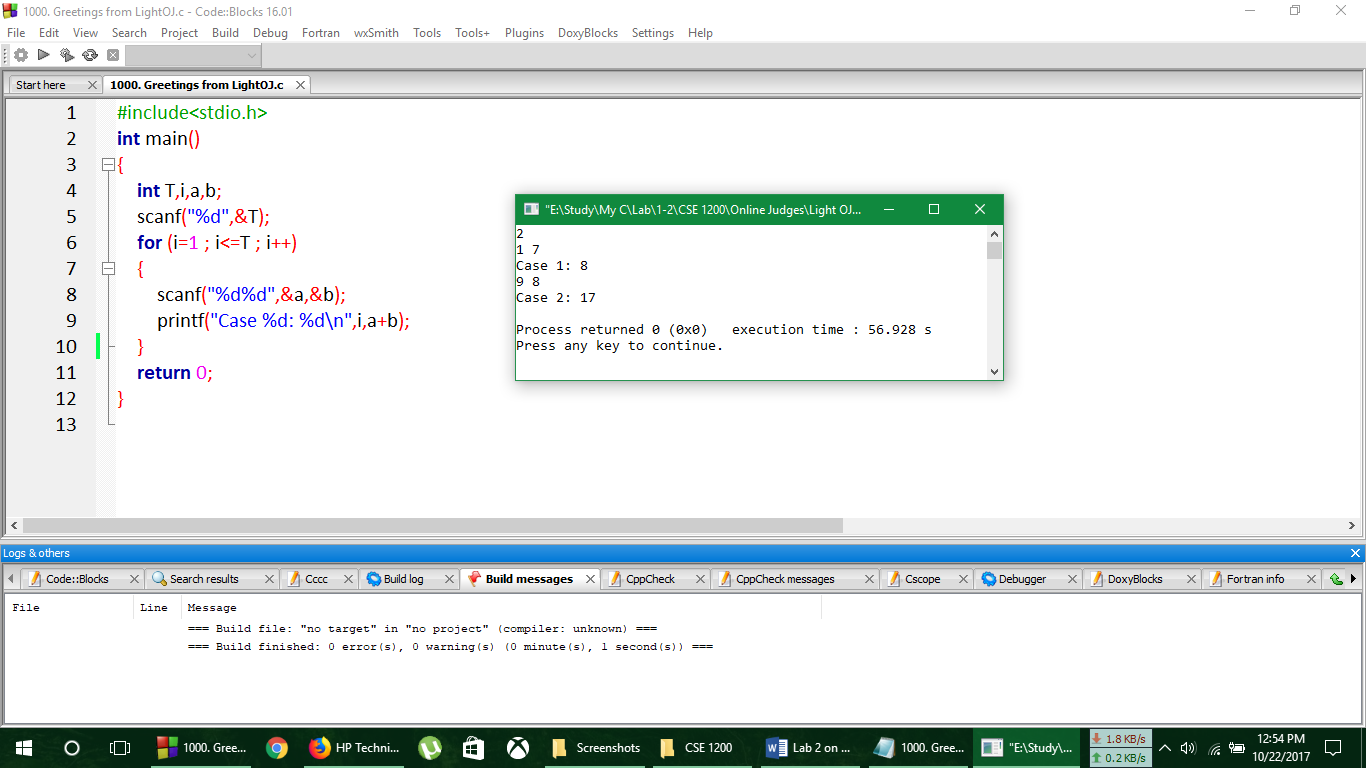
For each case, print the case number and the total number of problems. See the samples for exact formatting.

|  |  |
| --- | --- |
| Sample Input | Output for Sample Input |
| 2  1 7  9 8 | Case 1: 8  Case 2: 17 |

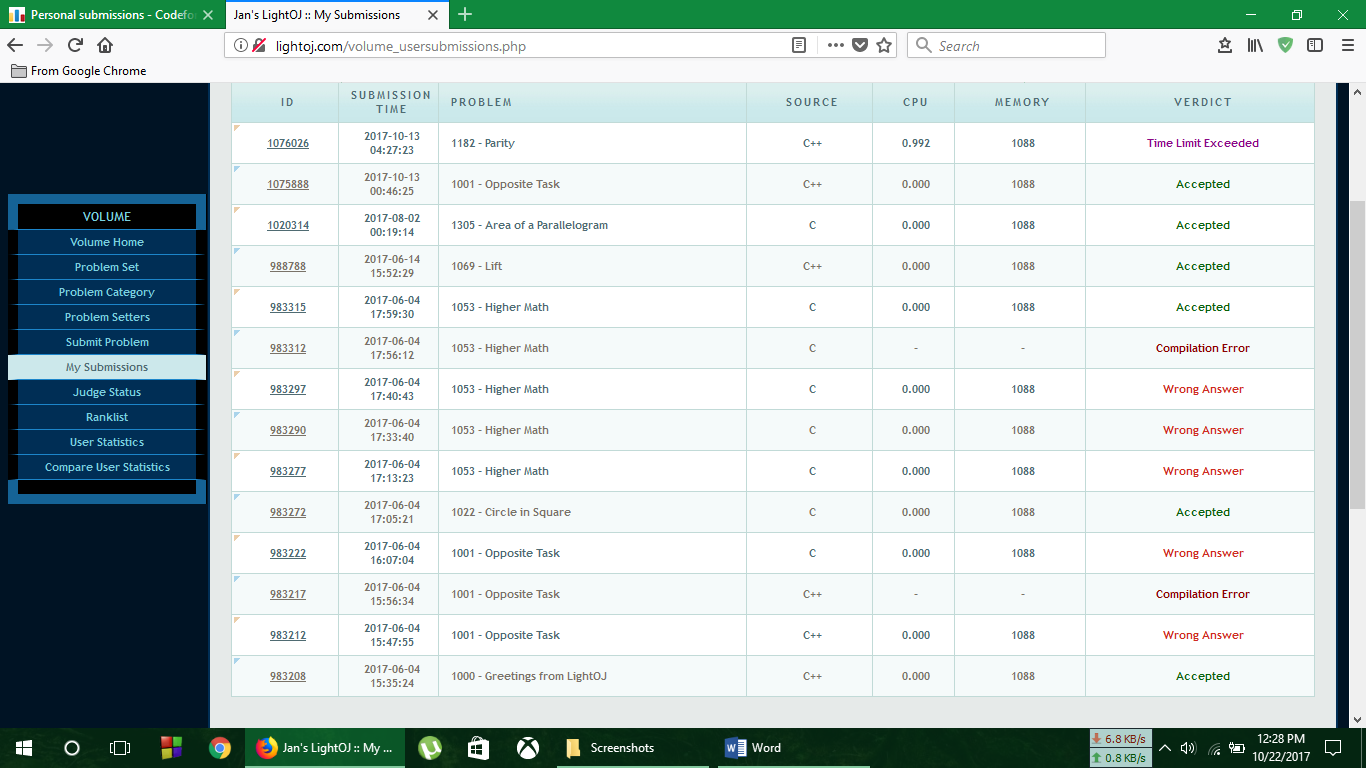
**Code**



**Output**



**Verdict**



**Analysis of Solution**

Here the complexity of the above program is O(T) .

**No. of Problem solved**: **7**

**Light Online Judge**

**Problem No. 1001**

**Opposite Task**

This problem gives you a flavor the concept of special judge. That means the judge is smart enough to verify your code even though it may print different results. In this problem you are asked to find the opposite task of the previous problem. To be specific, I have two computers where I stored my problems. Now I know the total number of problems is **n**. And there are no duplicate problems and there can be at most **10** problems in each computer. You have to find the number of problems in each of the computers. Since there can be multiple solutions. Any valid solution will do.

**Input**

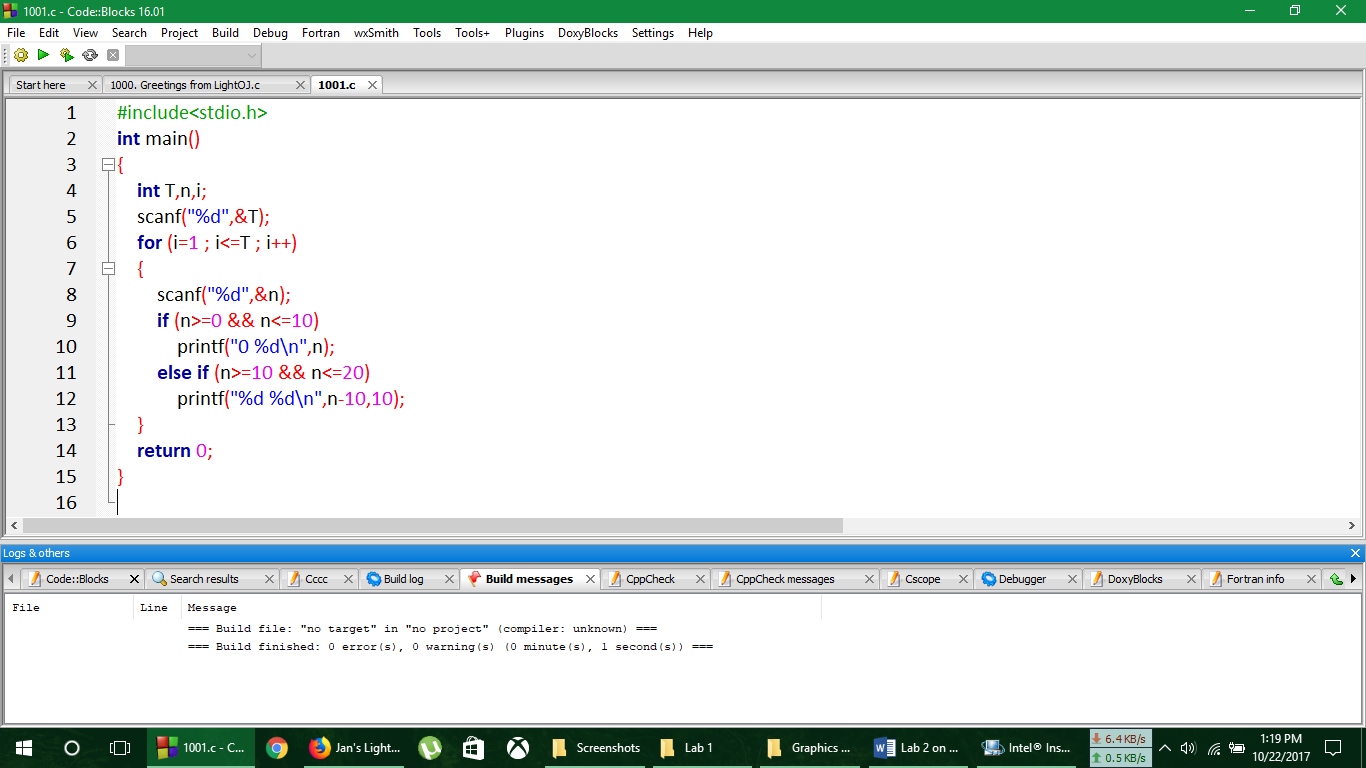
Input starts with an integer **T (≤ 25)**, denoting the number of test cases. Each case starts with a line containing an integer **n (0 ≤ n ≤ 20)** denoting the total number of problems.

**Output**

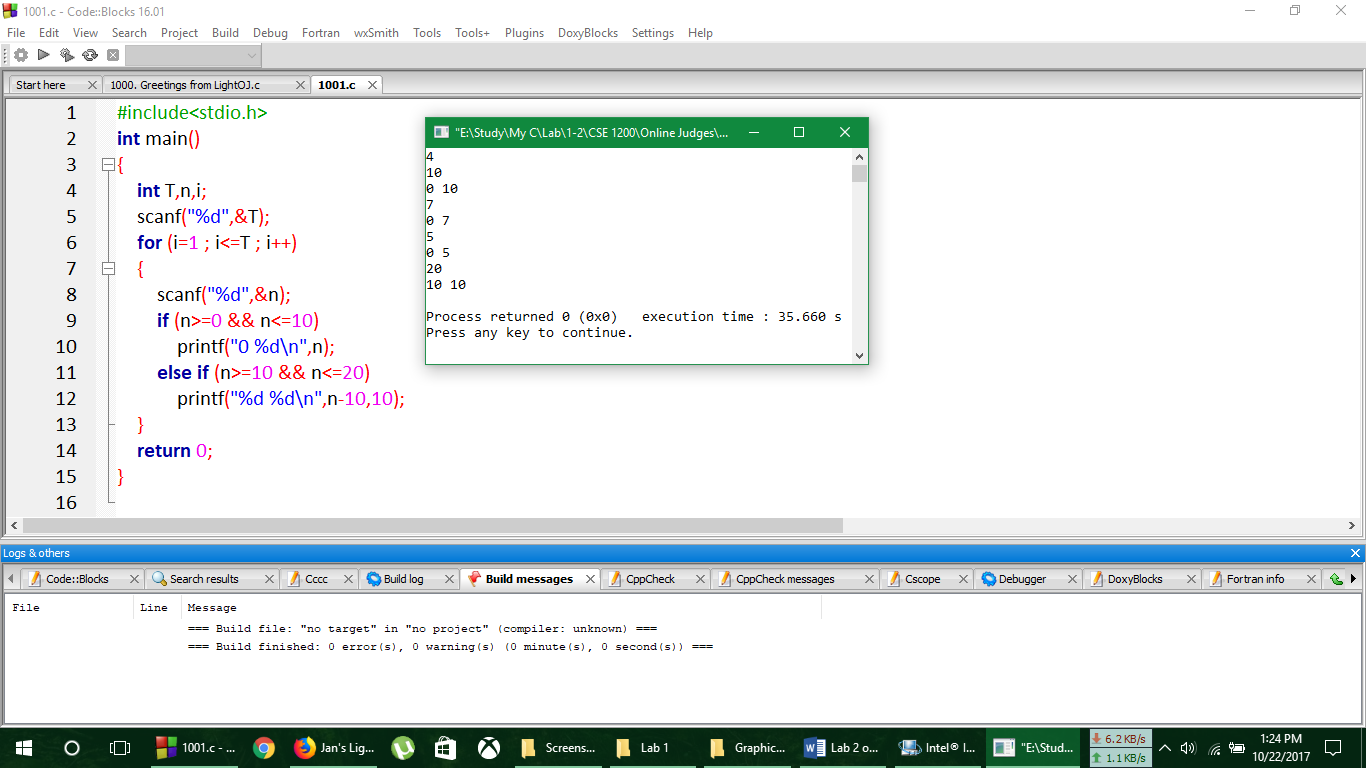
For each case, print the number of problems stored in each computer in a single line. A single space should separate the non-negative integers.

|  |  |
| --- | --- |
| Sample Input | Output for Sample Input |
| 3  10  7  7 | 0 10  0 7  1 6 |

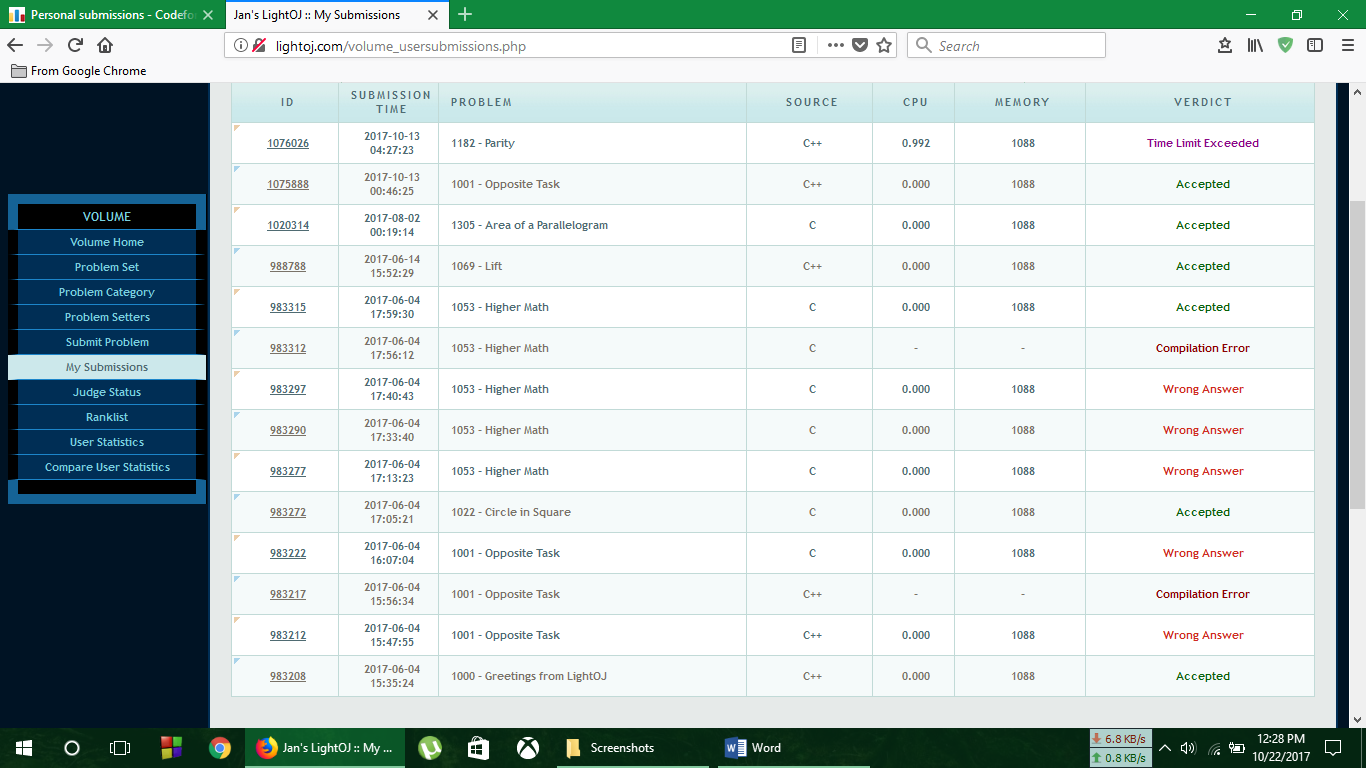
**Code**



**Output**



**Verdict**



**Analysis of Solution**

Here the complexity of the above program is O(T) .

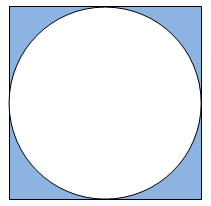
**No. of Problem solved**: **8**

**Light Online Judge**

**Problem No. 1022**

**Circle in Square**

A circle is placed perfectly into a square. The term perfectly placed means that each side of the square is touched by the circle, but the circle doesn't have any overlapping part with the square. See the picture below. Now you are given the radius of the circle. You have to find the area of the shaded region (blue part). Assume that pi = 2 \* acos (0.0) (acos means cos inverse).



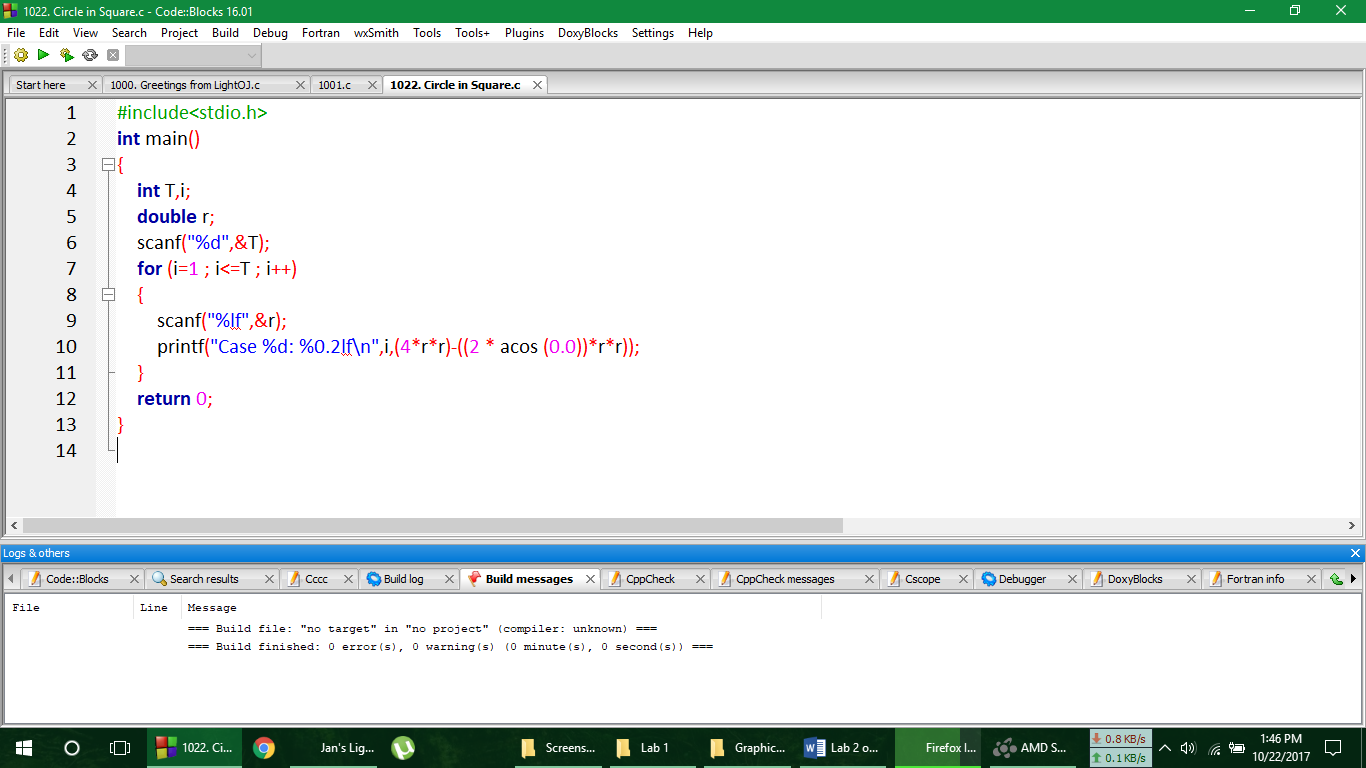
**Input**

Input starts with an integer T (≤ 1000), denoting the number of test cases. Each case contains a floating point number r (0 < r ≤ 1000) denoting the radius of the circle. And you can assume that r contains at most four digits after the decimal point.

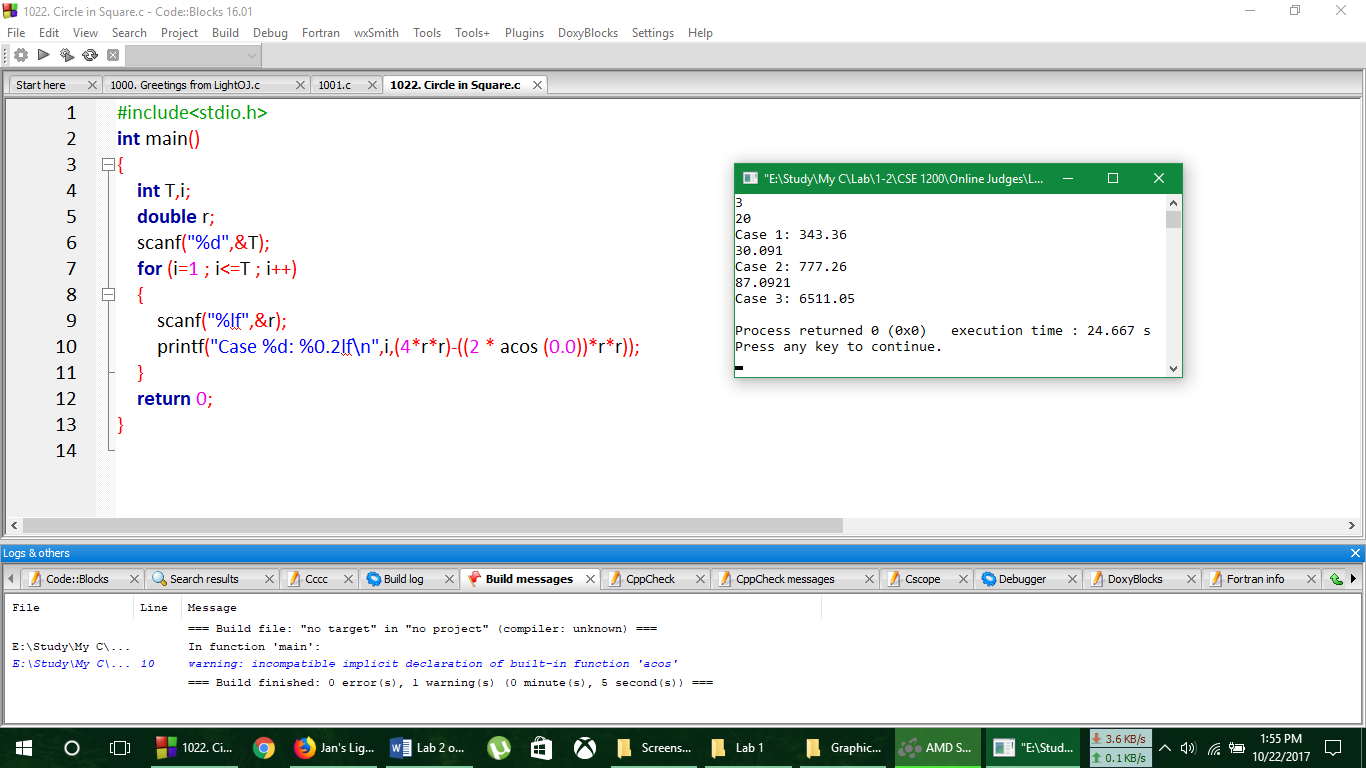
**Output**

For each case, print the case number and the shaded area rounded to two places after the decimal point.

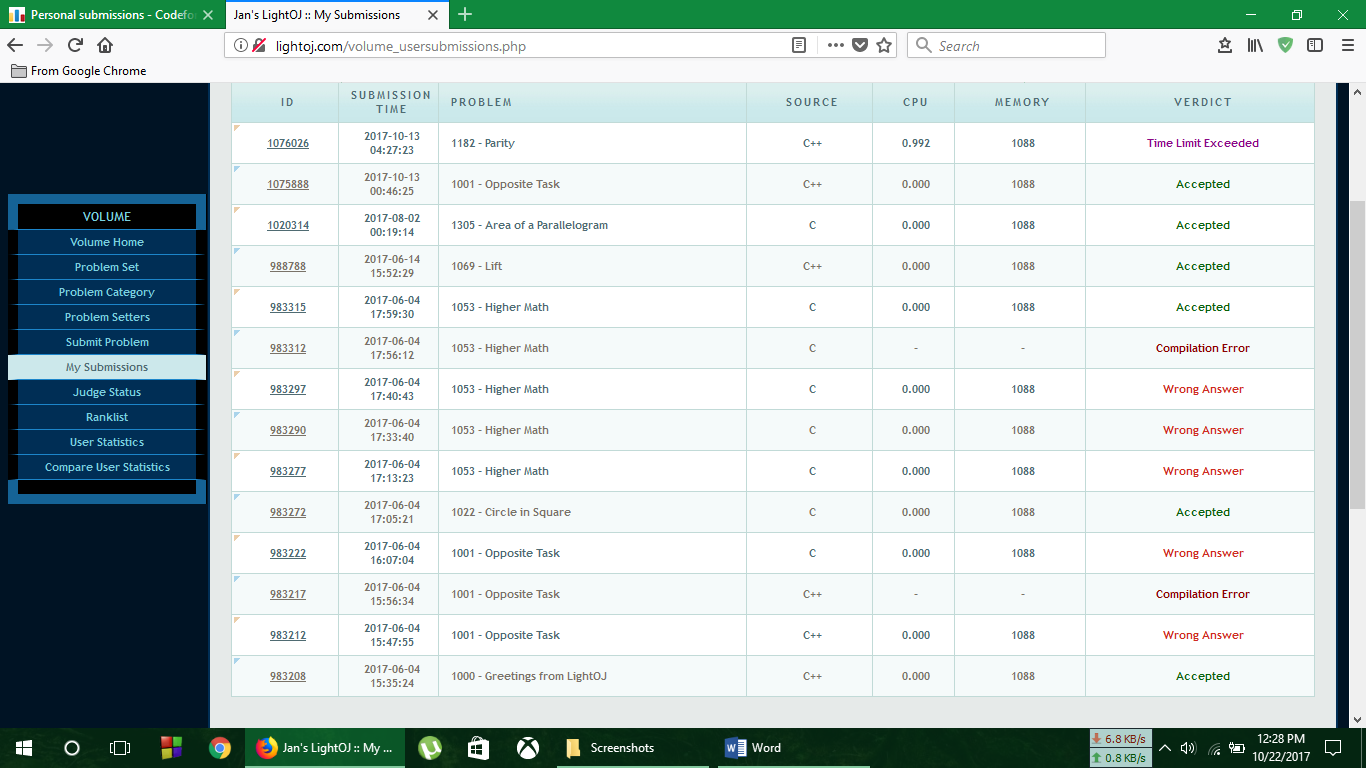
|  |  |
| --- | --- |
| Sample Input | Output for Sample Input |
| 3  20  30.091  87.0921 | Case 1: 343.36  Case 2: 777.26  Case 3: 6511.05 |

**Code**

**Output**



**Verdict**



**Analysis of Solution**

Here the complexity of the above program is O(T) .

**No. of Problem solved**: **9**

**Light Online Judge**

**Problem No. 1053**

**Higher Math**

You are building a house. You'd prefer if all the walls have a precise right angle relative to the ground, but you have no device to measure angles. A friend says he has a great idea how you could ensure that all walls are upright: All you need to do is step away a few feet from the wall, measure how far away you are from the wall, measure the height of the wall, and the distance from the upper edge of the wall to where you stand. You friend tells you to do these measurements for all walls, then he'll tell you how to proceed. Sadly, just as you are done, a timber falls on your friend, and an ambulance brings him to the hospital. This is too bad, because now you have to figure out what to do with your measurements yourself. Given the three sides of a triangle, determine if the triangle is a right triangle, i.e. if one of the triangle's angles is 90 degrees.

**Input**

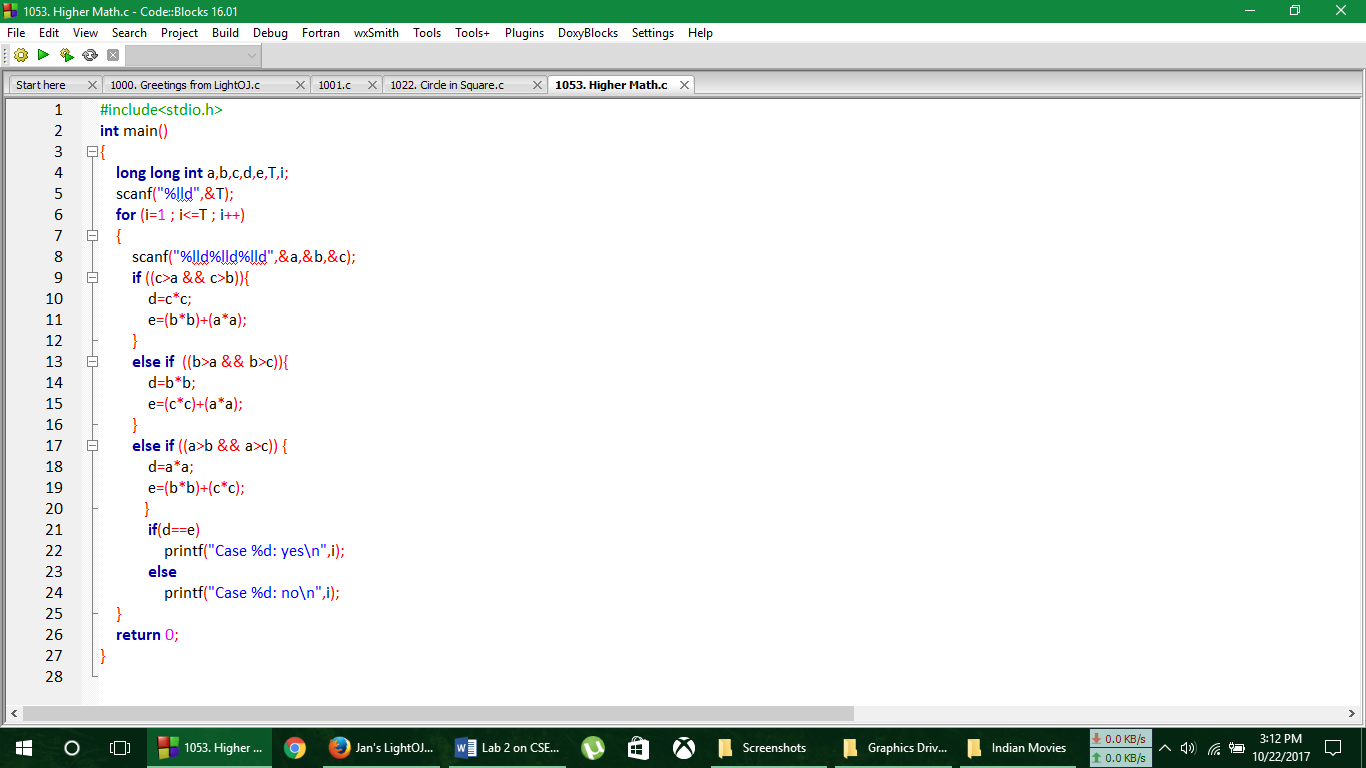
Input starts with an integer T (≤ 200), denoting the number of test cases. Each test case consists of three integers 1 ≤ a, b, c ≤ 40000 separated by a space. The three integers are the lengths of the sides of a triangle.

**Output**

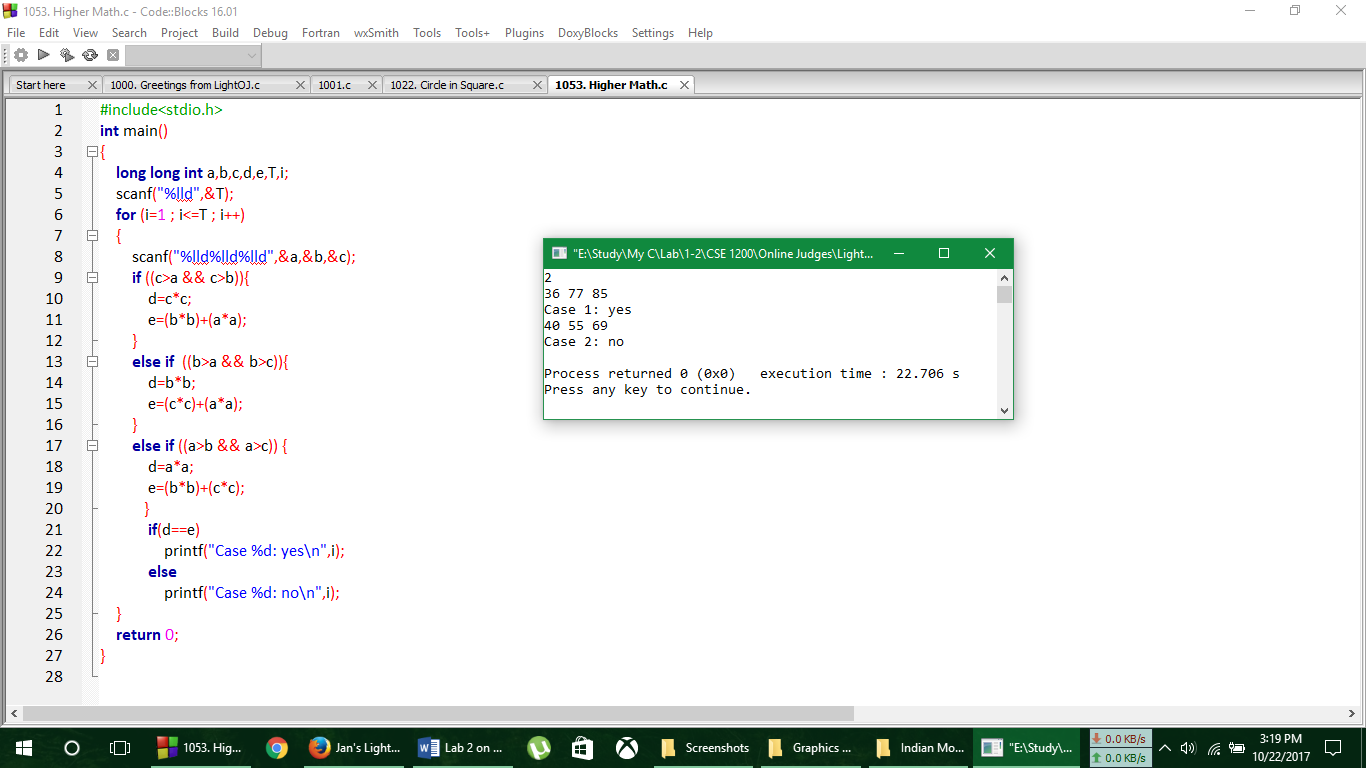
For each case, print the case number and "yes" or "no" depending on whether it's a right angle or not.

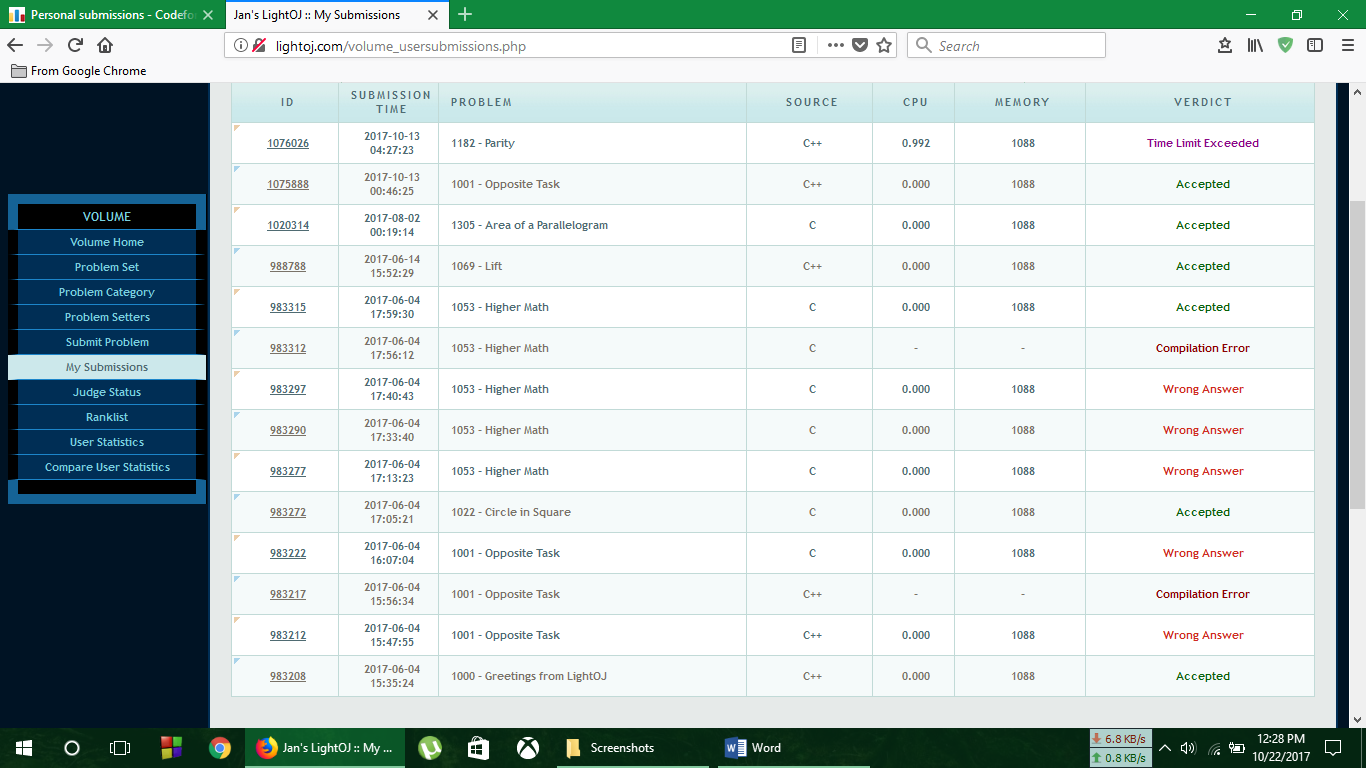
|  |  |
| --- | --- |
| Sample Input | Output for Sample Input |
| 2  36 77 85  40 55 69 | Case 1: yes  Case 2: no |

**Code**



**Output**



**Verdict**

**Analysis of Solution**

Here the complexity of the above code is O(T) .

**No. of Problem solved**: 10

**Light Online Judge**

**Problem No. 1069**

**Lift**

All of you must have noticed that the lift of AIUB is not available for students. But since you deny obeying usual rules, you always use this lift no matter what happens! Now one day you were sleeping in the class and when you woke up you found none in the department except the guard who was in a deep sleep in his room. But luckily you found the lift on. So, you want to go to the ground floor using the lift. But the lift can be in a different floor. Then you must wait for the lift to come to your floor. Assume that it takes 4 seconds for the lift to go from any floor to its adjacent floor (up or down). It takes 3 seconds to open or close the door of the lift and you need 5 seconds to enter or exit the lift. Given your position and the lift's position you have to calculate the time for you to reach the ground floor (floor 0). Reaching ground floor means you must get out of the lift in ground floor.

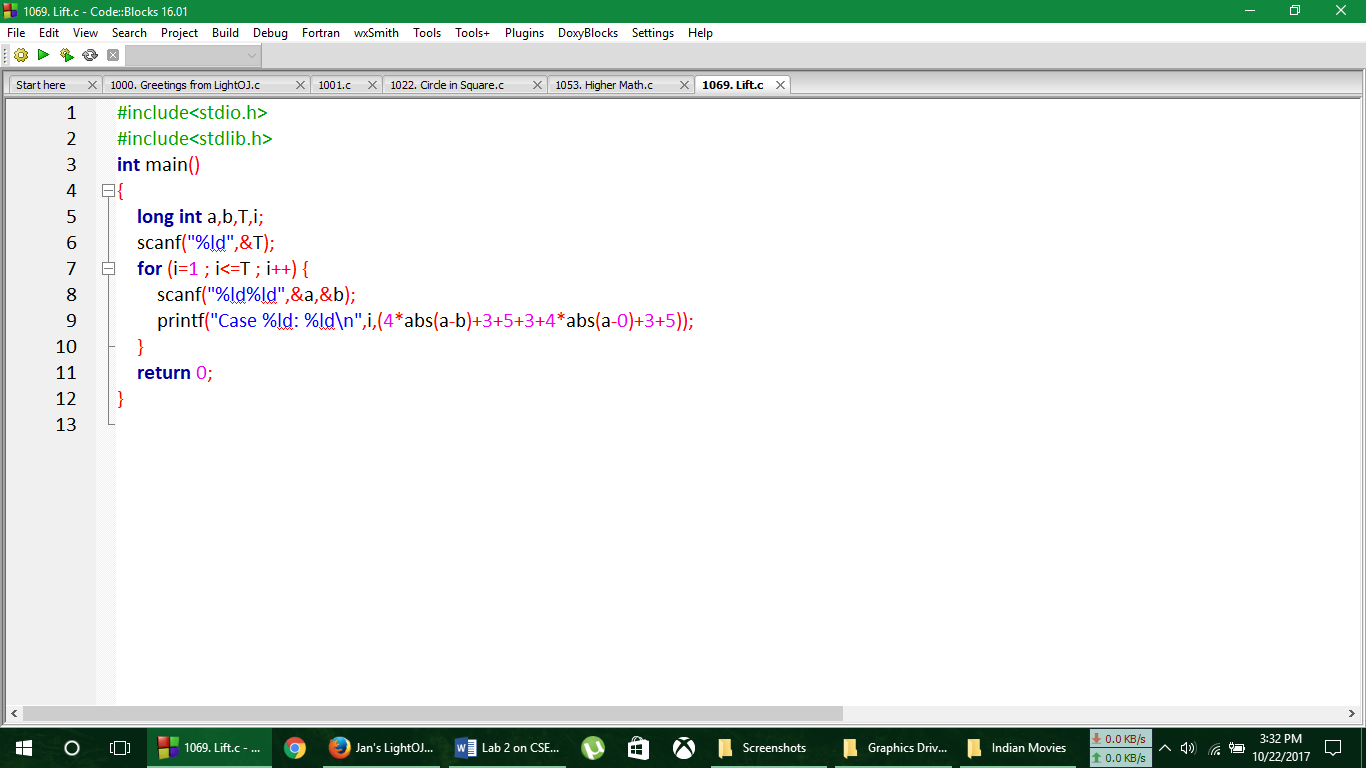
**Input**

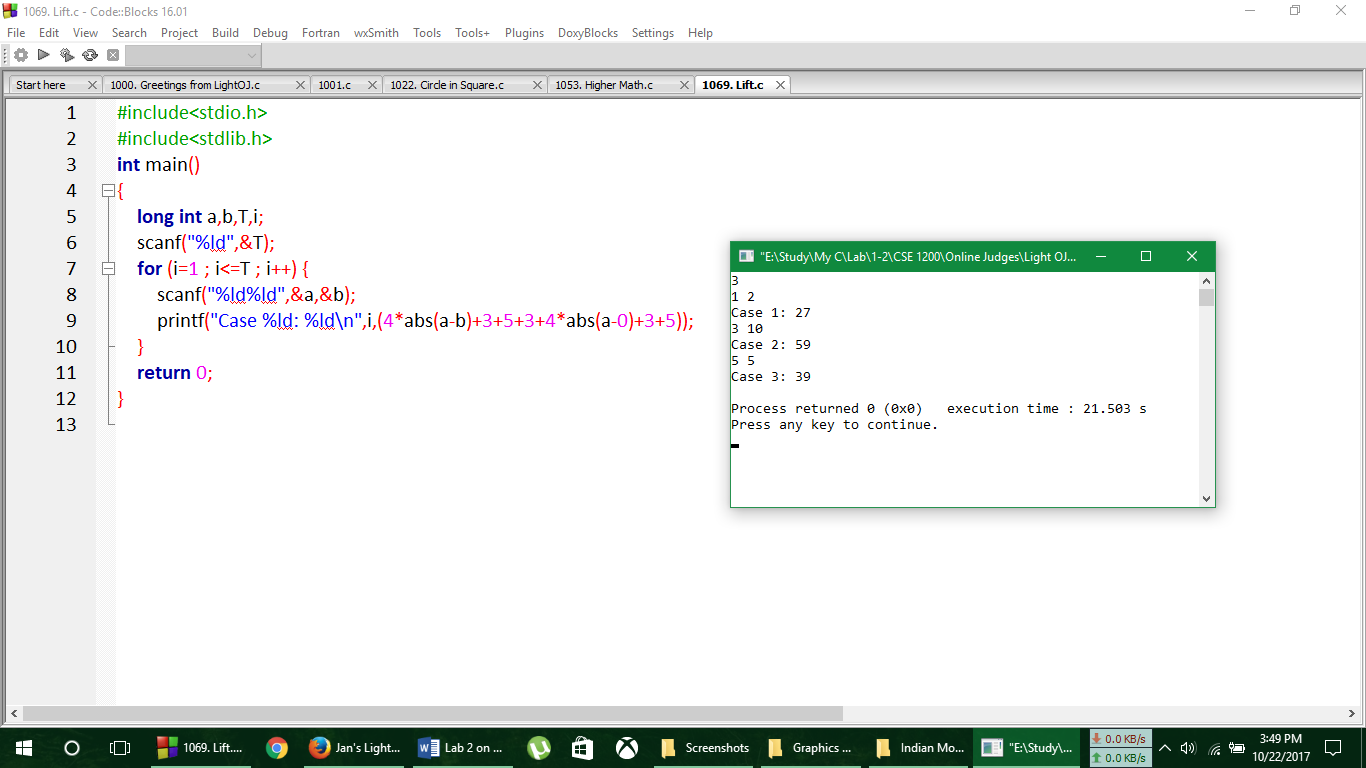
Input starts with an integer T (≤ 25), denoting the number of test cases. Each case contains two integers. The first integer means your position (different than 0) and the second integer means the position of the lift. Assume that the department has 100 floors (may be one day it will be :D).

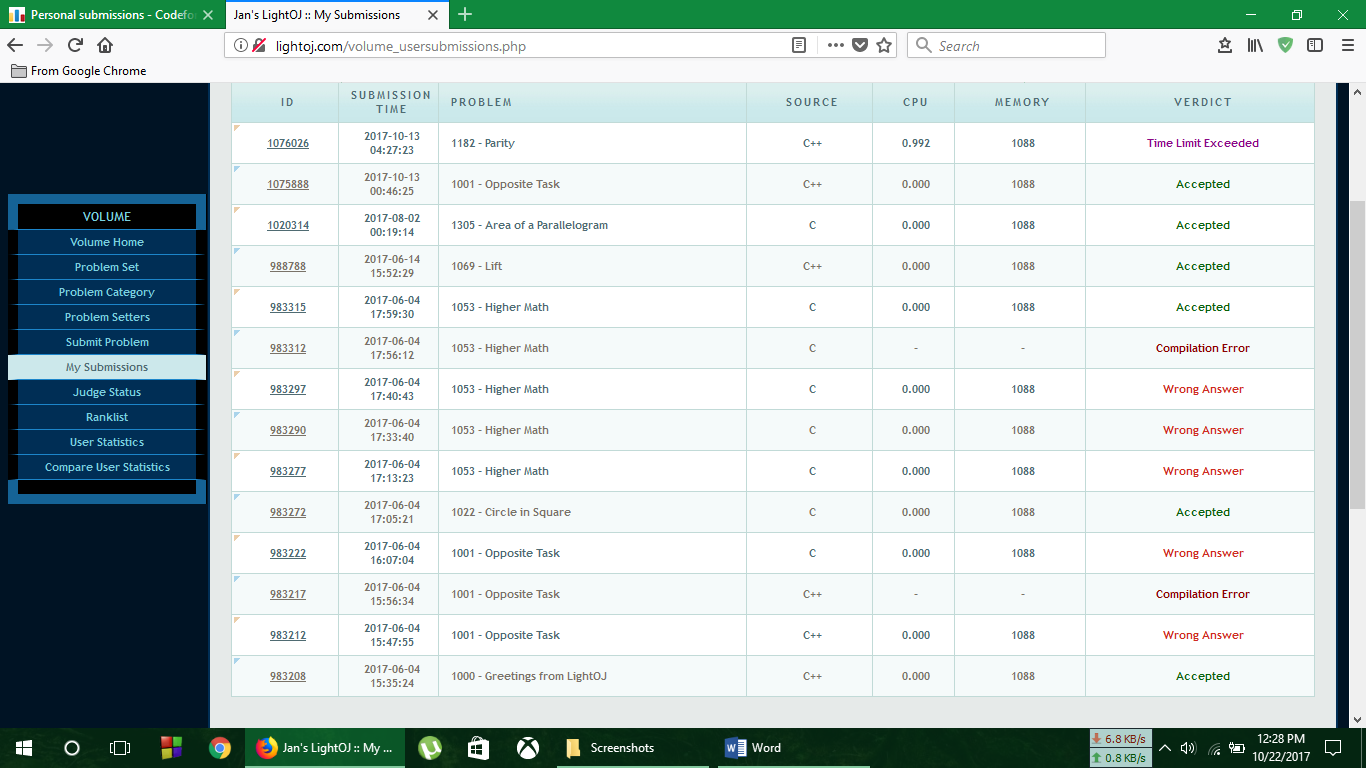
**Output**

For each case, print the case number and the calculated time in seconds.

|  |  |
| --- | --- |
| Sample Input | Output for Sample Input |
| 3  1 2  3 10  5 5 | Case 1: 27  Case 2: 59  Case 3: 39 |

**Code**

**Output**

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(T) .

**No. of Problem solved**: 11

**Light Online Judge**

**Problem No. 1305**

**Area of a Parallelogram**

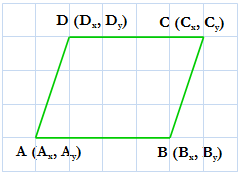
A parallelogram is a quadrilateral with two pairs of parallel sides. Now you are given the co-ordinates of A, B and C, you have to find the coordinates of D and the area of the parallelogram. The orientation of ABCD should be same as in the picture.

Fig: a parallelogram **Input**

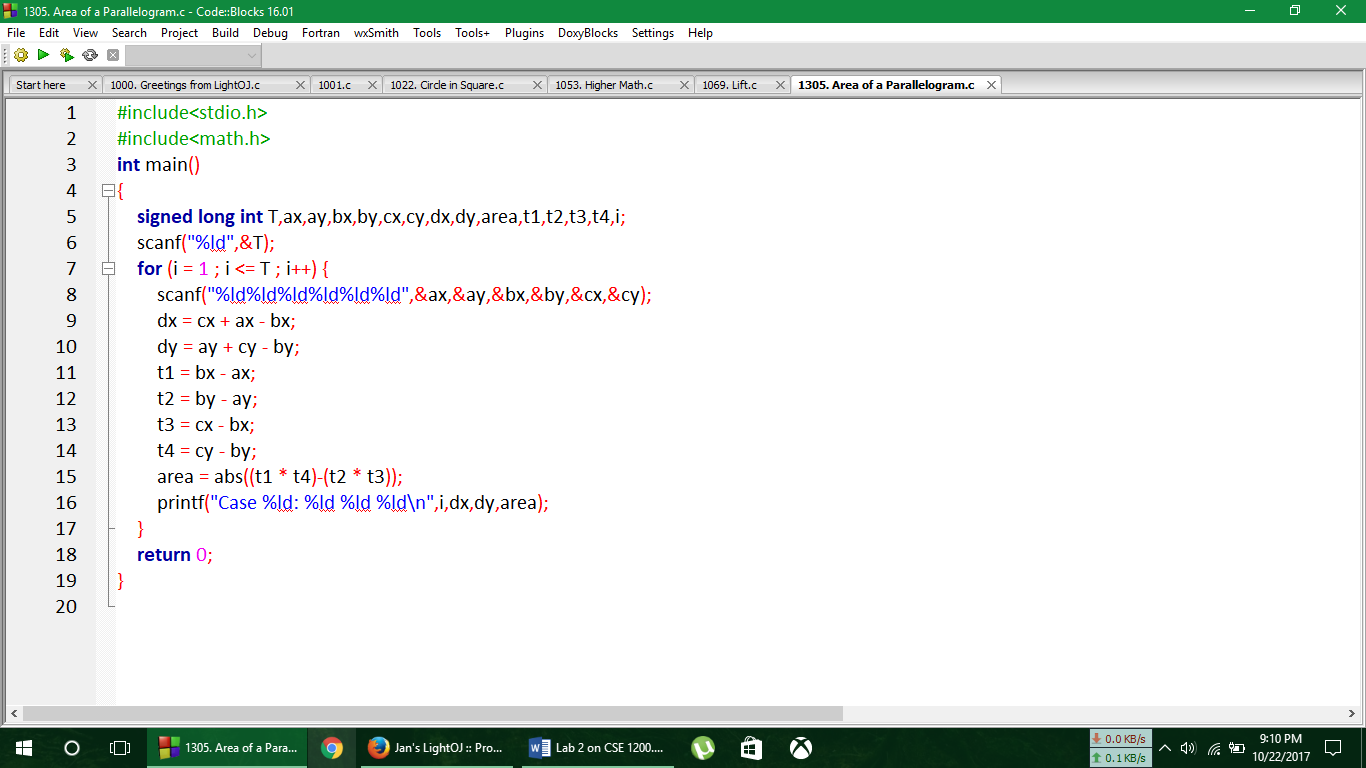
Input starts with an integer T (≤ 1000), denoting the number of test cases. Each case starts with a line containing six integers Ax, Ay, Bx, By, Cx, Cy where (Ax, Ay) denotes the coordinate of A, (Bx, By) denotes the coordinate of B and (Cx, Cy) denotes the coordinate of C. Value of any coordinate lies in the range [-1000, 1000]. And you can assume that A, B and C will not be collinear.

**Output**

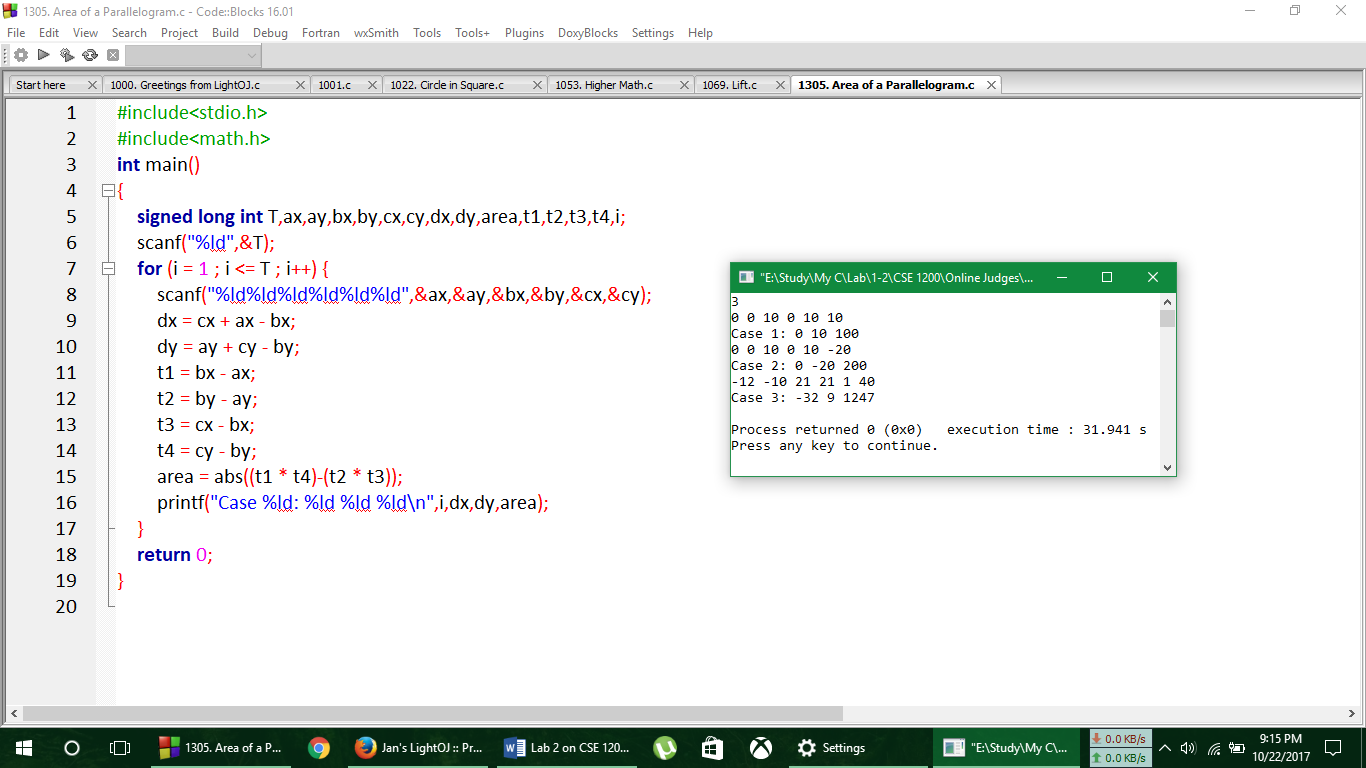
For each case, print the case number and three integers where the first two should be the coordinate of D and the third one should be the area of the parallelogram.

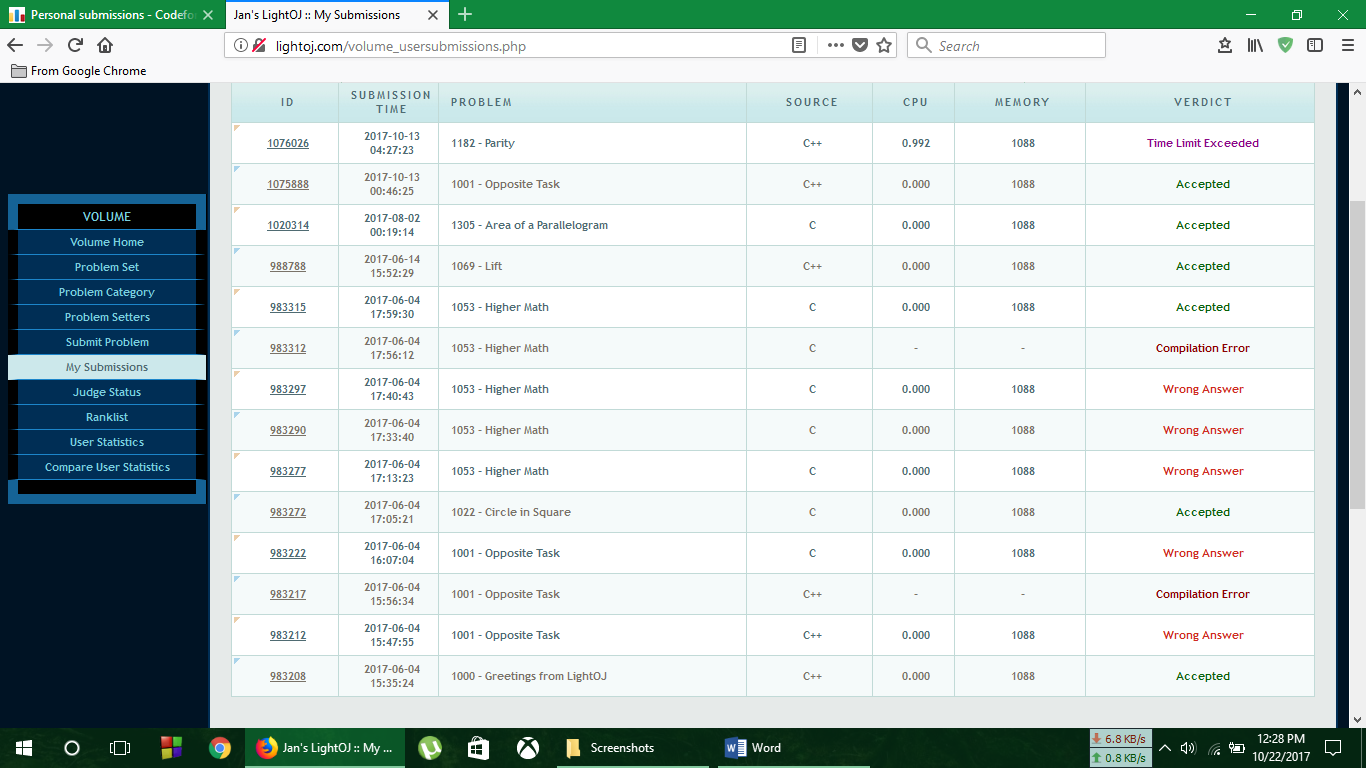
|  |  |
| --- | --- |
| Sample Input | Output for Sample Input |
| 3  0 0 10 0 10 10  0 0 10 0 10 -20  -12 -10 21 21 1 40 | Case 1: 0 10 100  Case 2: 0 -20 200  Case 3: -32 9 1247 |

**Code**



**Output**

****

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(T) .

**No. of Problem solved**: 12

**UVa Online Judge**

**Problem No. p10071**

**Back to High School Physics**

A particle has initial velocity and acceleration. If its velocity after certain time is v then what will its displacement be in twice of that time?

**Input**

The input will contain two integers in each line. Each line makes one set of input. These two integers. Denote the value of v ( -100 < v < 100) and t (0 < t < 200) (t means at the time the particle gains that velocity)

**Output**

For each line of input print a single integer in one line denoting the displacement in double of that time.

**Sample Input**

0 0

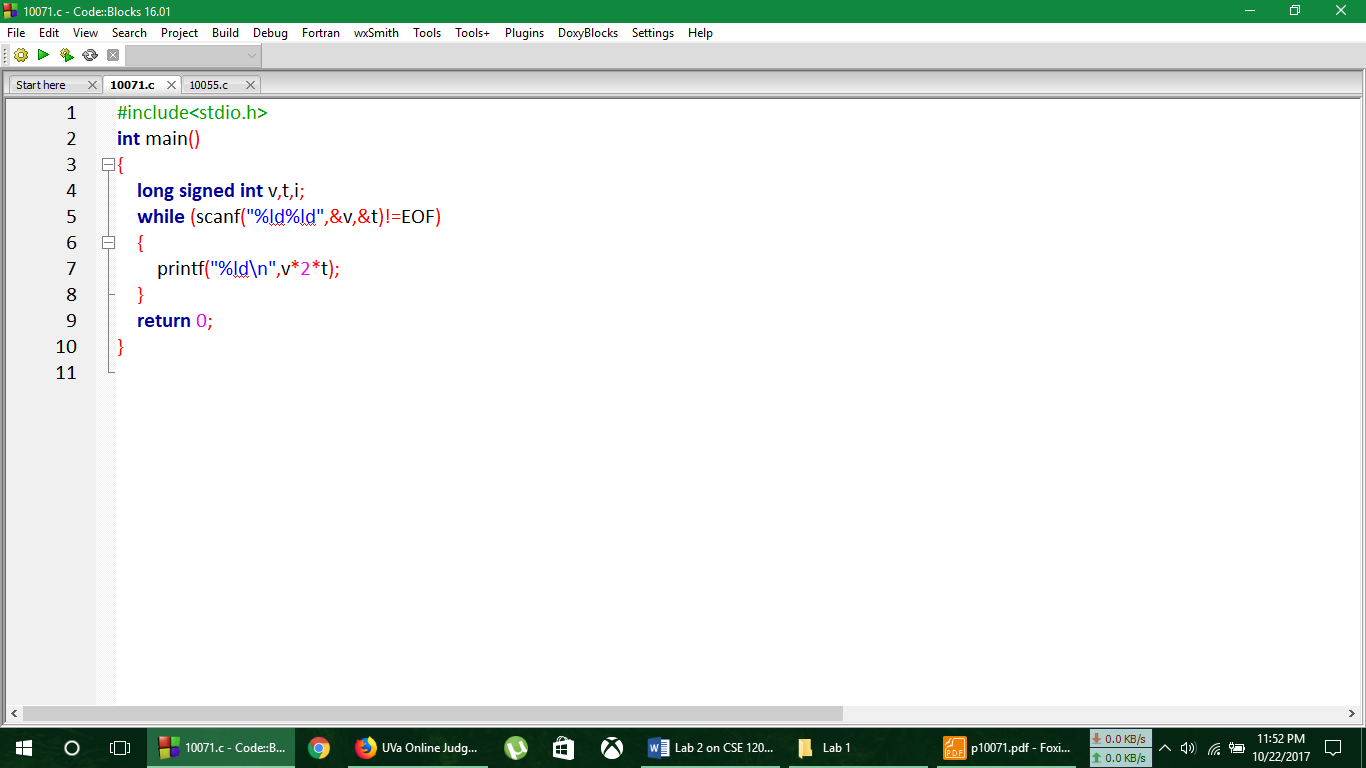
5 12

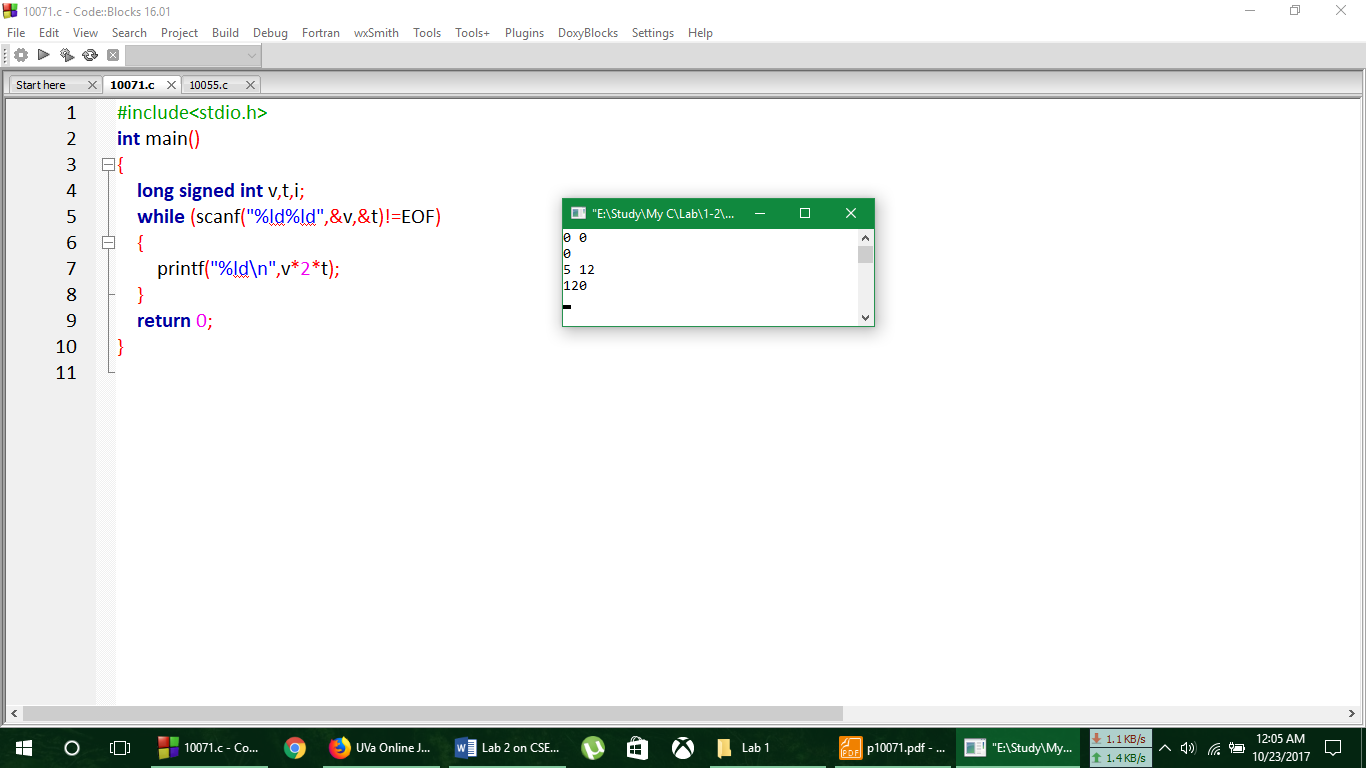
**Sample Output**

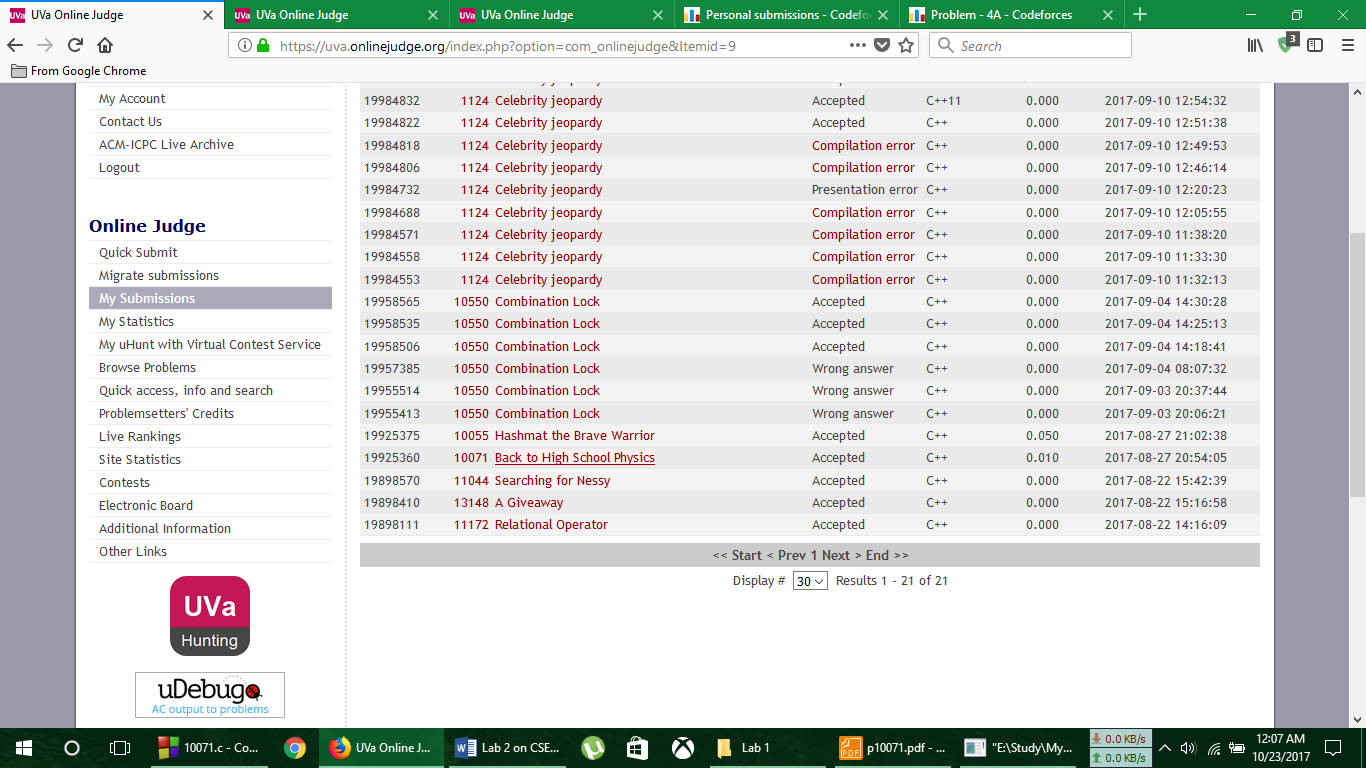
0

120

**Code**



**Output**

**Verdict**

**Analysis of Solution**

Here the complexity of the program is O(EOF)

**No. of Problem solved**: 13

**UVa Online Judge**

**Problem No. 10055**

**Hashmat the Brave Warrior**

Hashmat is a brave warrior who with his group of young soldiers moves from one place to another to fight against his opponents. Before Fighting he just calculates one thing, the difference between his soldier number and the opponent's soldier number. From this difference he decides whether to fight or not. Hashmat's soldier number is never greater than his opponent.

**Input**

The input contains two numbers in every line. These two numbers in each line denotes the number soldiers in Hashmat's army and his opponent's army or vice versa. The input numbers are not greater than 2^(32). Input is terminated by `End of File'.

**Output**

For each line of input, print the difference of number of soldiers between Hashmat's army and his opponent's army. Each output should be in seperate line.

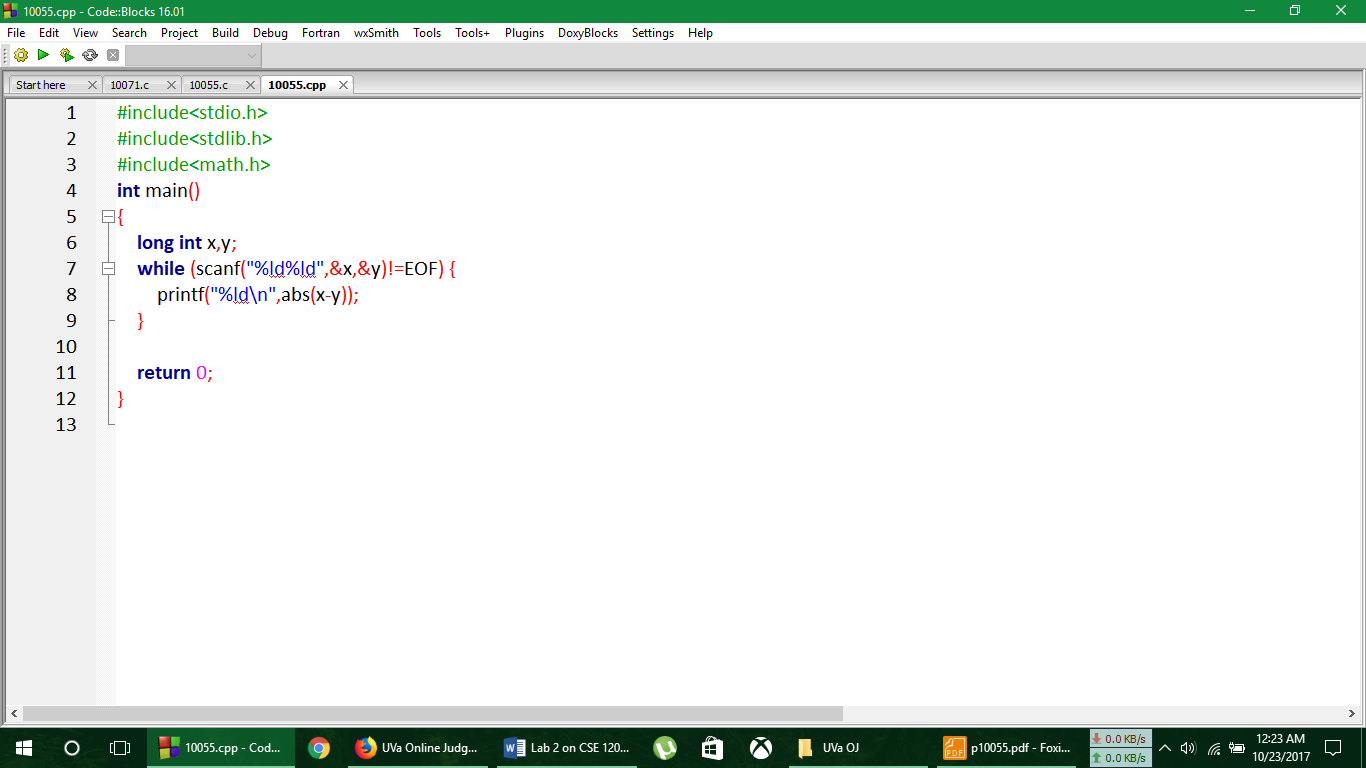
**Sample Input Sample Output**

10 12 2

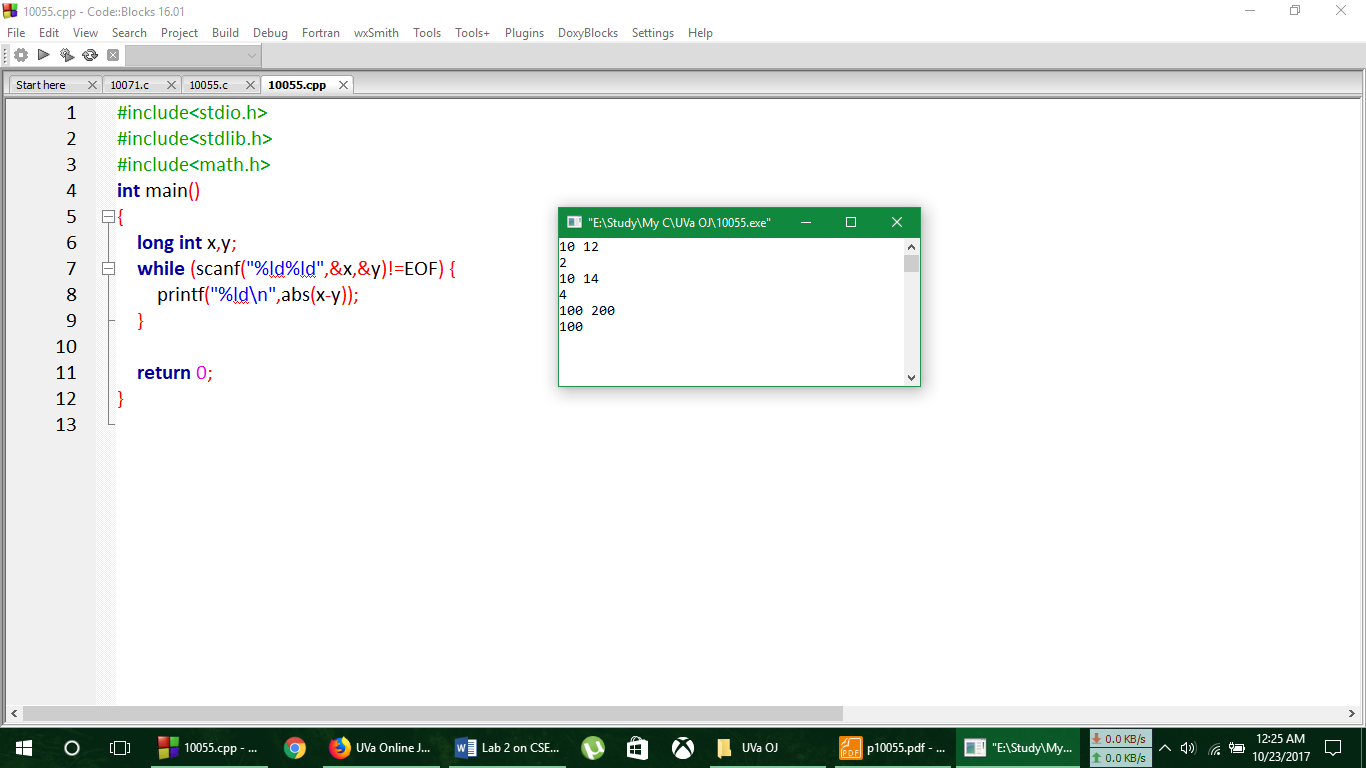
10 14 4

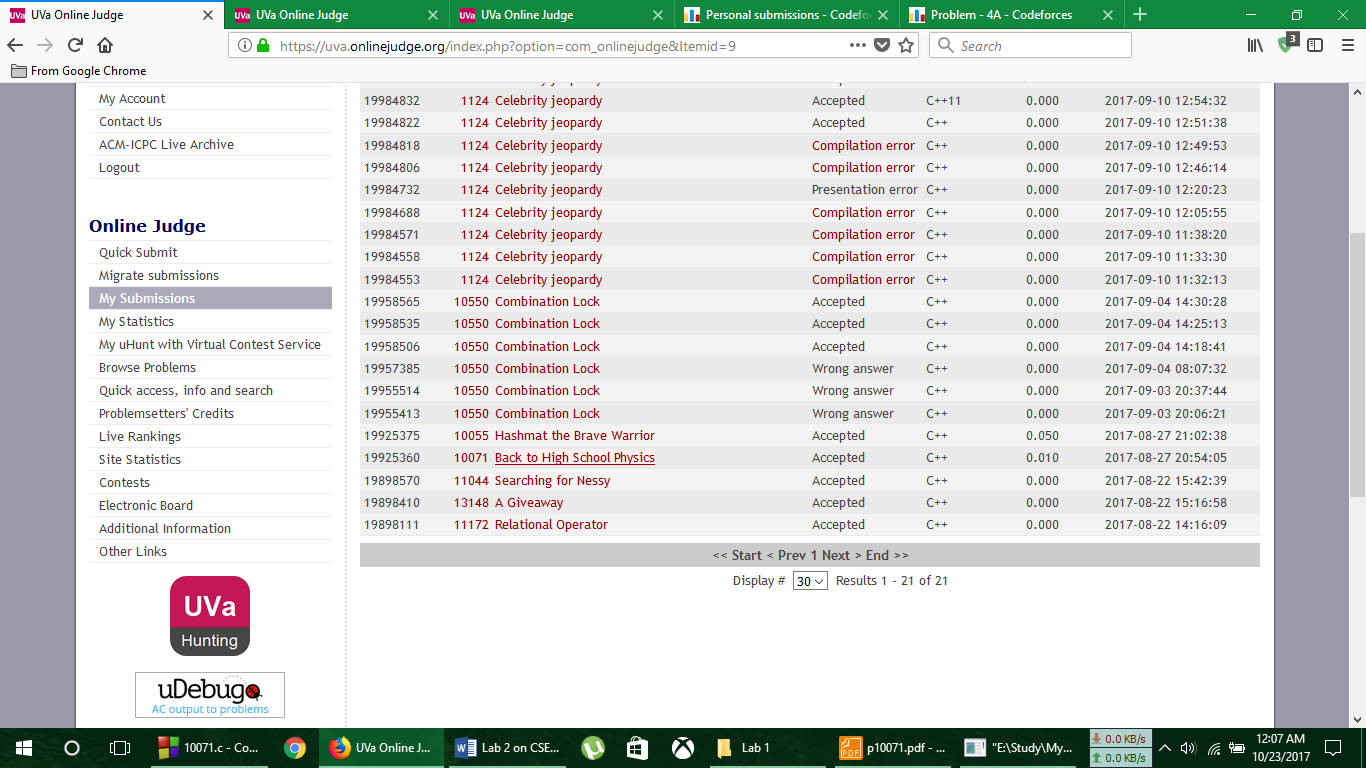
100 200 100

**Code**

****

**Output**



**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(EOF) .

**No. of Problem solved**: 14

**Codeforces Online Judge**

**Problem No. 4A**

**Watermelon**

One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed *w* kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem. Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

**Input**

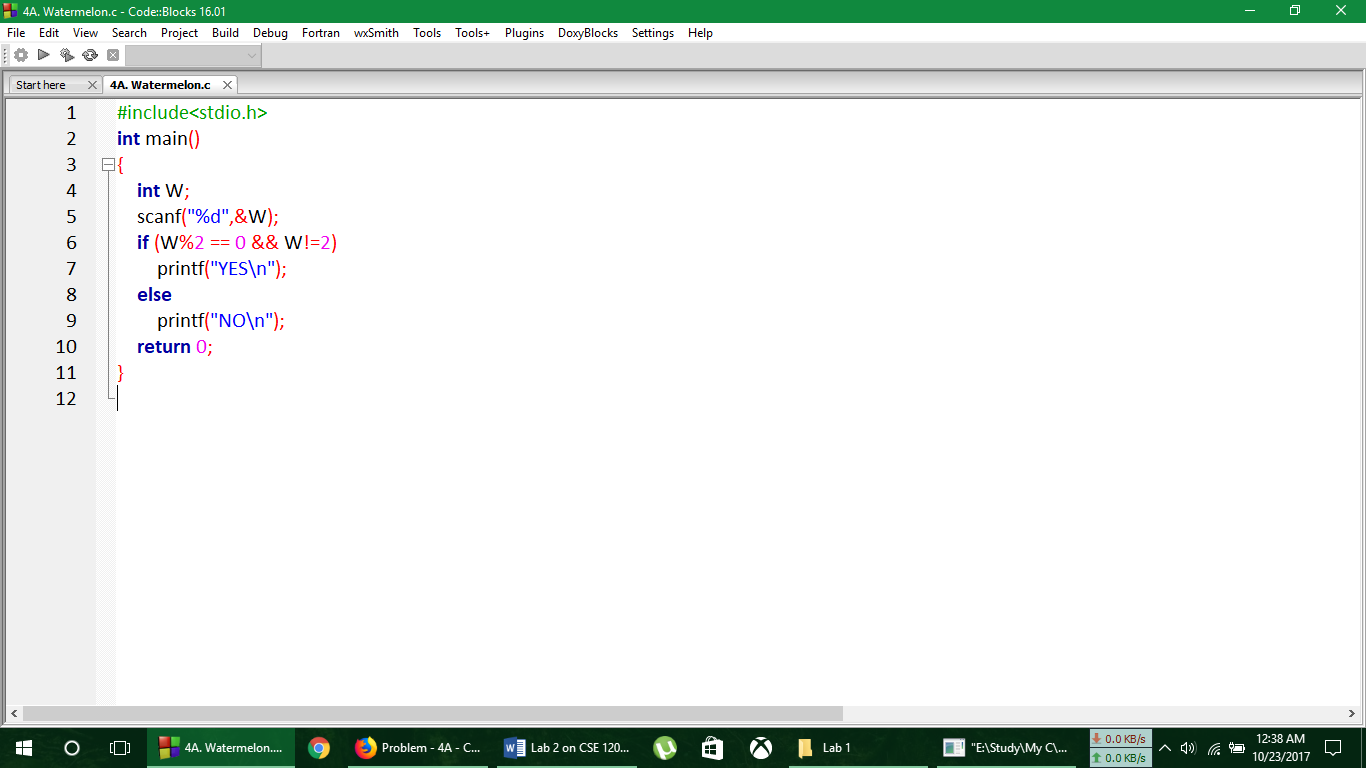
The first (and the only) input line contains integer number *w* (1 ≤ *w* ≤ 100) — the weight of the watermelon bought by the boys.

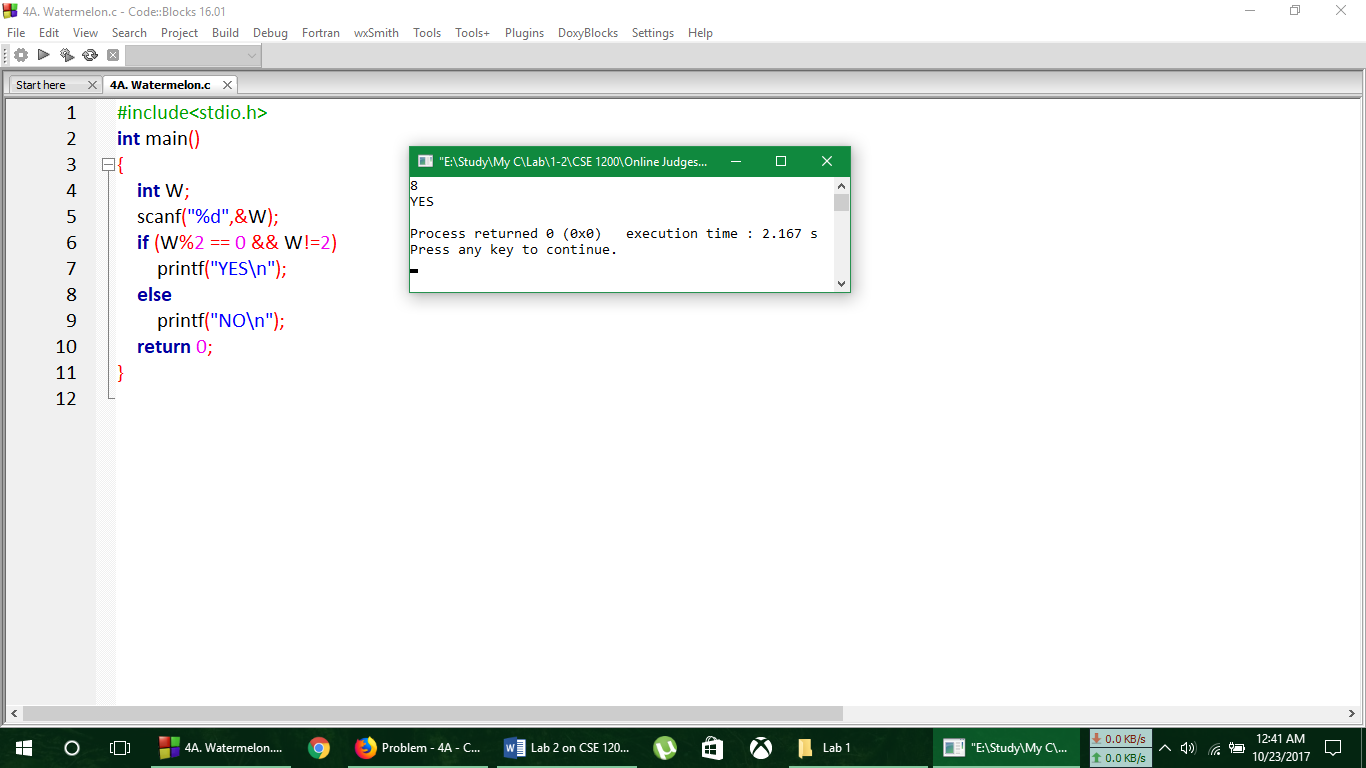
**Output**

Print YES, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and NO in the opposite case.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 8 | YES |

**Code**



**Output**

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(1)

**No. of Problem solved**: 15

**Codeforces Online Judge**

**Problem No. 71A**

**Way Too Long Words**

Sometimes some words like "localization" or "internationalization" are so long that writing them many times in one text is quite tiresome. Let's consider a word too long, if its length is strictly more than 10 characters. All too long words should be replaced with a special abbreviation. This abbreviation is made like this: we write down the first and the last letter of a word and between them we write the number of letters between the first and the last letters. That number is in decimal system and doesn't contain any leading zeroes. Thus, "localization" will be spelt as "l10n", and "internationalization» will be spelt as "i18n". You are suggested to automatize the process of changing the words with abbreviations. At that all too long words should be replaced by the abbreviation and the words that are not too long should not undergo any changes.

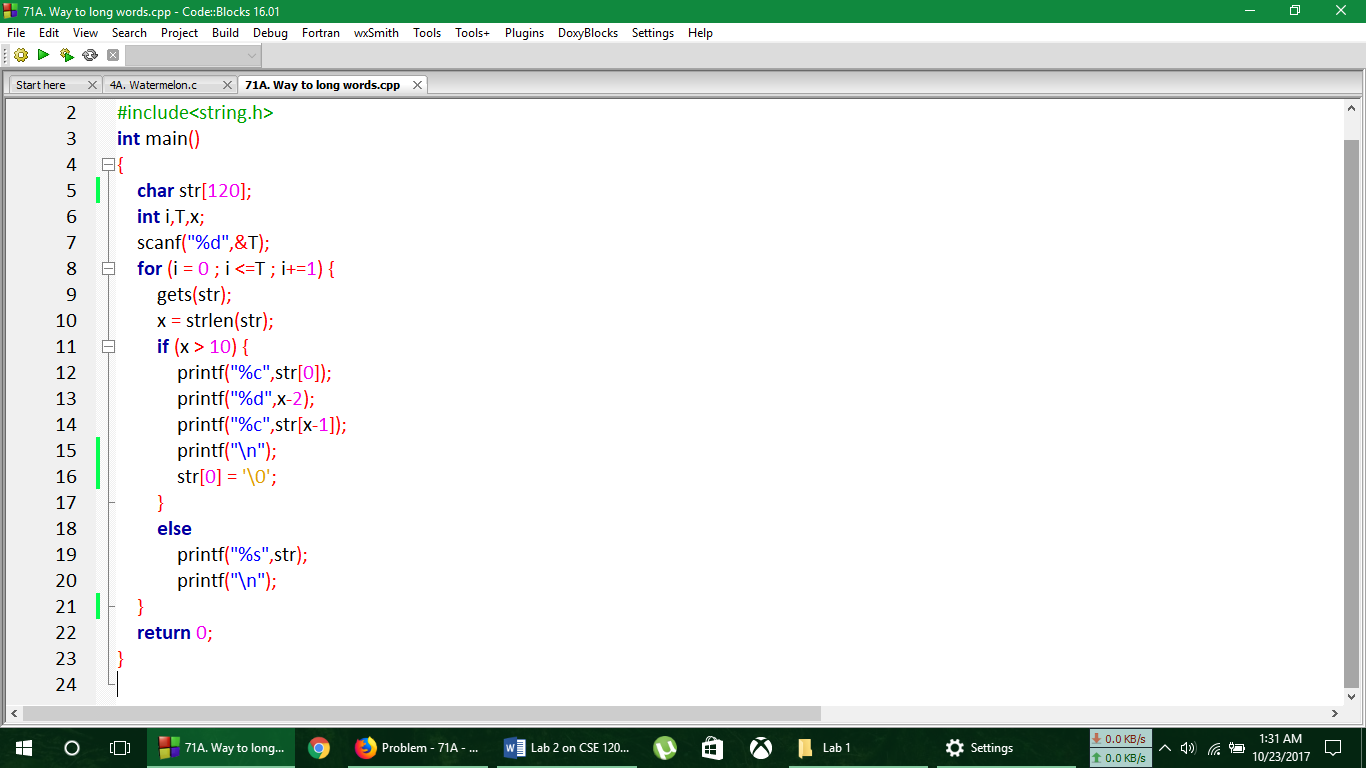
**Input**

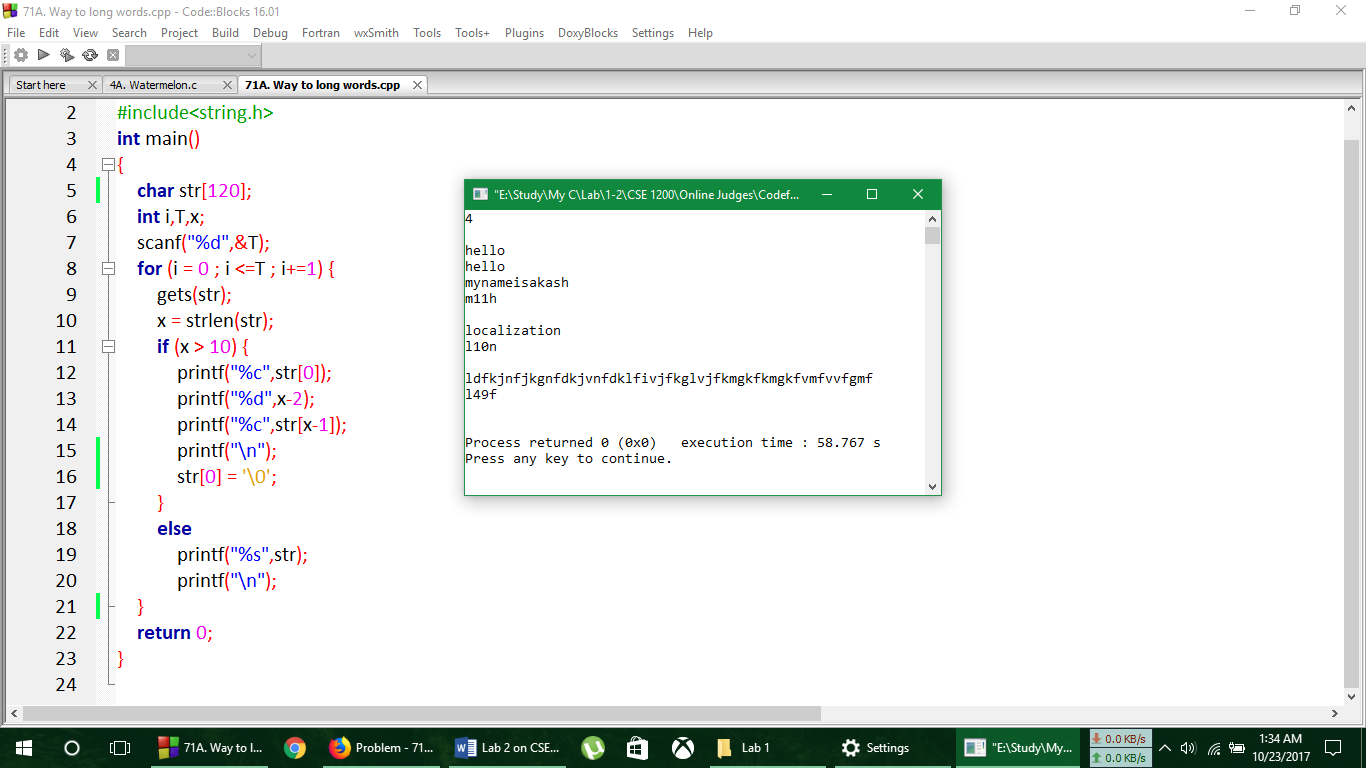
The first line contains an integer *n* (1 ≤ *n* ≤ 100). Each of the following *n* lines contains one word. All the words consist of lowercase Latin letters and possess the lengths of from 1 to 100 characters.

**Output**

Print *n* lines. The *i*-th line should contain the result of replacing of the *i*-th word from the input data.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 4  word  localization  internationalization  pneumonoultramicroscopicsilicovolcanoconiosis | word  l10n  i18n  p43s |

**Code**

**Output**

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(T) .

**No. of Problem solved**: 16

**Codeforces Online Judge**

**Problem No. 50A**

**Domino piling**

You are given a rectangular board of *M* × *N* squares. Also you are given an unlimited number of standard domino pieces of 2 × 1 squares. You are allowed to rotate the pieces. You are asked to place as many dominoes as possible on the board so as to meet the following conditions:

1. Each domino completely covers two squares.

2. No two dominoes overlap.

3. Each domino lies entirely inside the board.

It is allowed to touch the edges of the board. Find the maximum number of dominoes, which can be placed under these restrictions.

**Input**

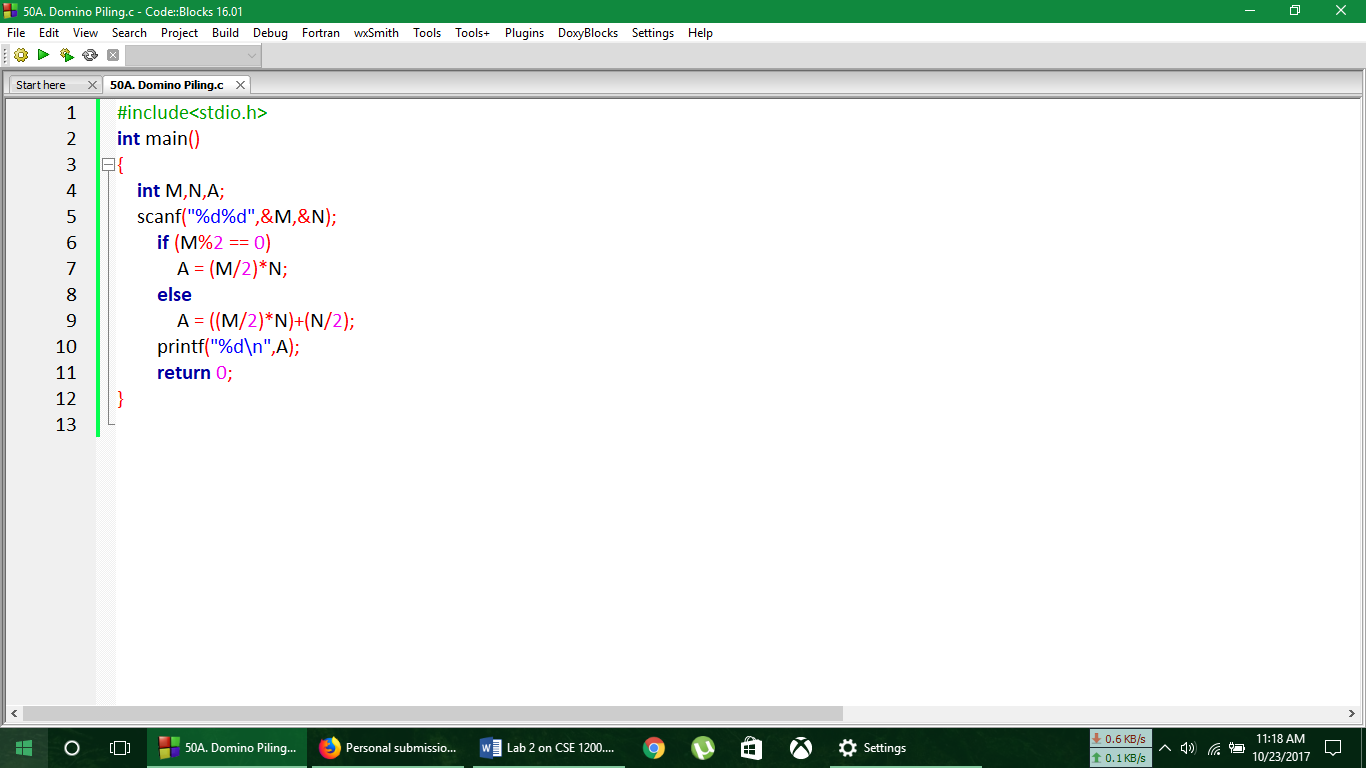
In a single line you are given two integers *M* and *N* — board sizes in squares (1 ≤ *M* ≤ *N* ≤ 16).

**Output**

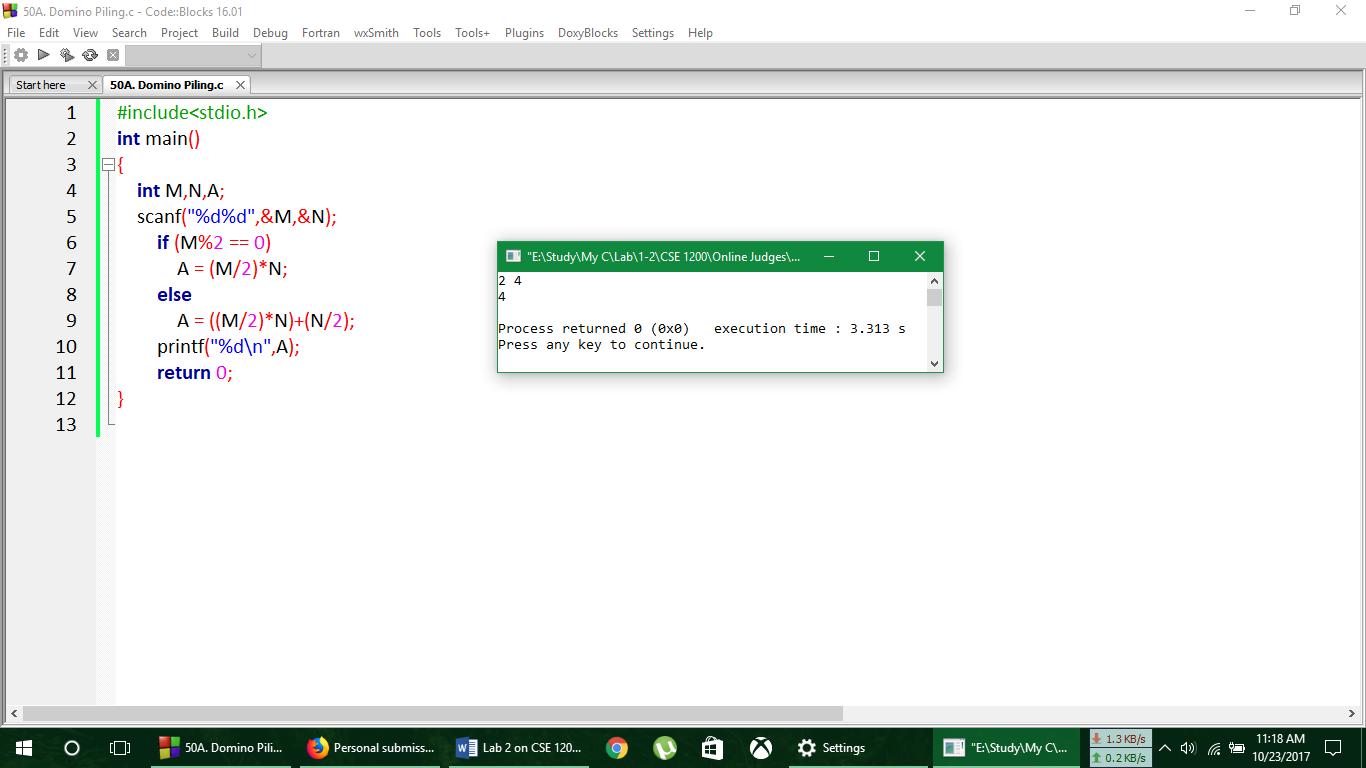
Output one number — the maximal number of dominoes, which can be placed.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 2 4 | 4 |
| 3 3 | 4 |

**Code**



**Output**

****

**Verdict**

****

**Analysis of Solution**

Here the complexity of the above program is O(1) .

**No. of Problem solved**: 17

**Codeforces Online Judge**

**Problem No. 546A**

**Soldier and Bananas**

A soldier wants to buy *w* bananas in the shop. He has to pay *k* dollars for the first banana, 2*k* dollars for the second one and so on (in other words, he has to pay *i*·*k* dollars for the *i*-th banana). He has *n* dollars. How many dollars does he have to borrow from his friend soldier to buy *w* bananas?

**Input**

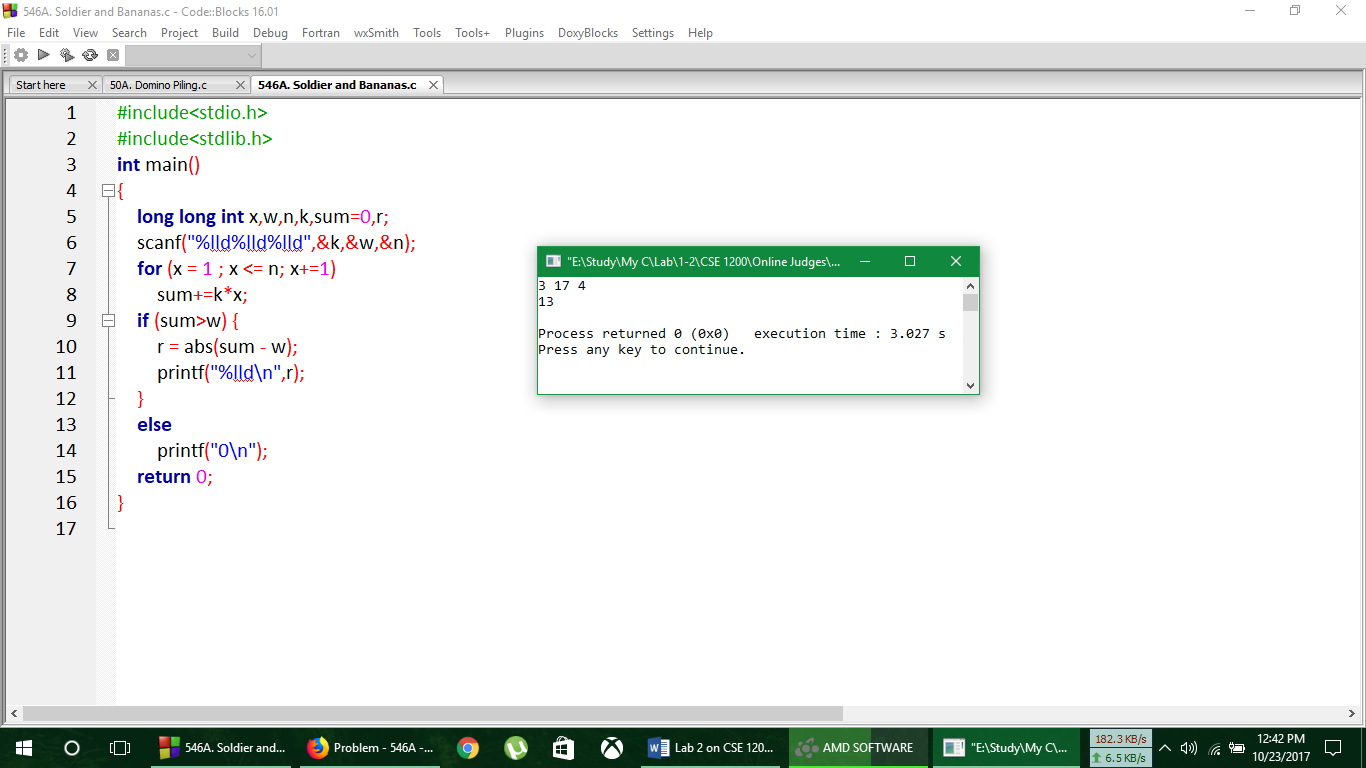
The first line contains three positive integers *k*, *n*, *w* (1  ≤  *k*, *w*  ≤  1000, 0 ≤ *n* ≤ 109), the cost of the first banana, initial number of dollars the soldier has and number of bananas he wants.

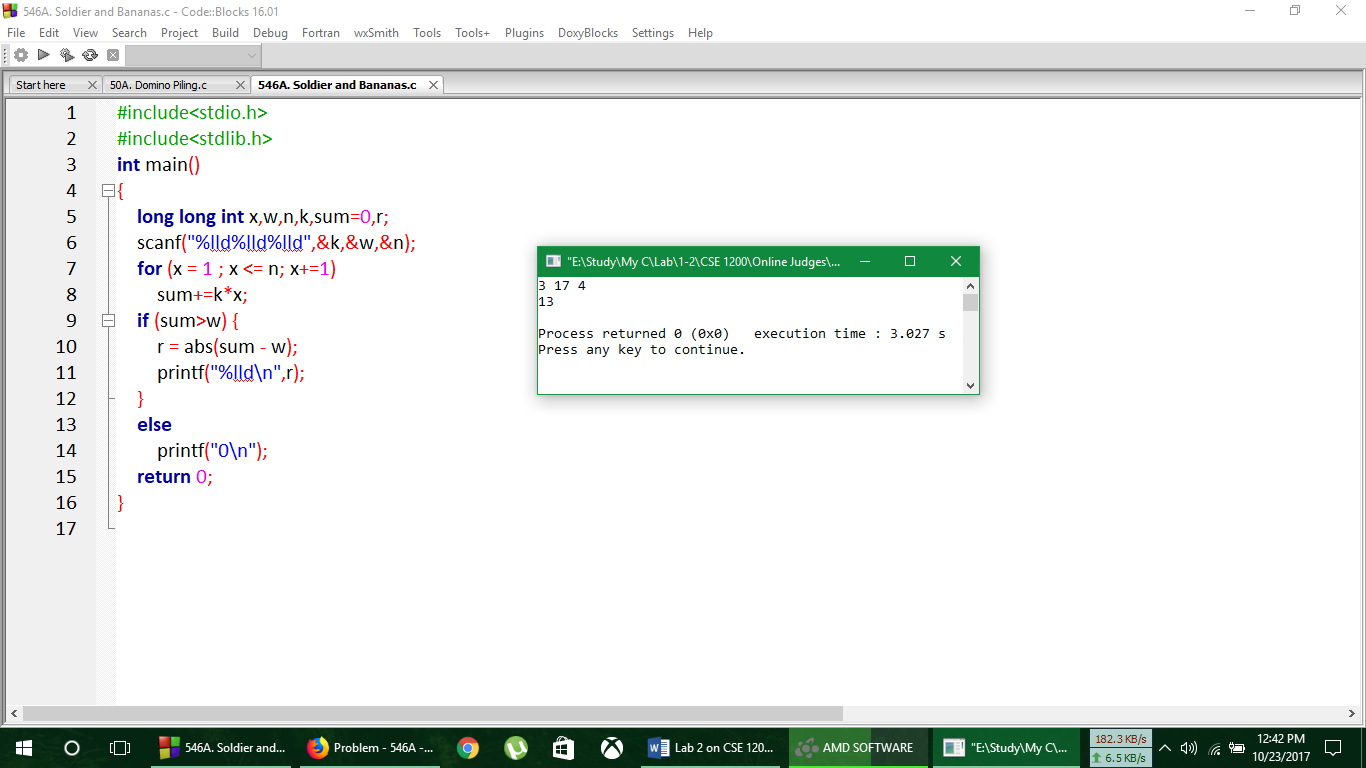
**Output**

Output one integer — the amount of dollars that the soldier must borrow from his friend. If he doesn't have to borrow money, output 0.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 3 17 4 | 13 |

**Code**

****

**Output**

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(n) .

**No. of Problem solved**: 18

**Codeforces Online Judge**

**Problem No. 281A**

**Word Capitalization**

Capitalization is writing a word with its first letter as a capital letter. Your task is to capitalize the given word. Note, that during capitalization all the letters except the first one remains unchanged.

**Input**

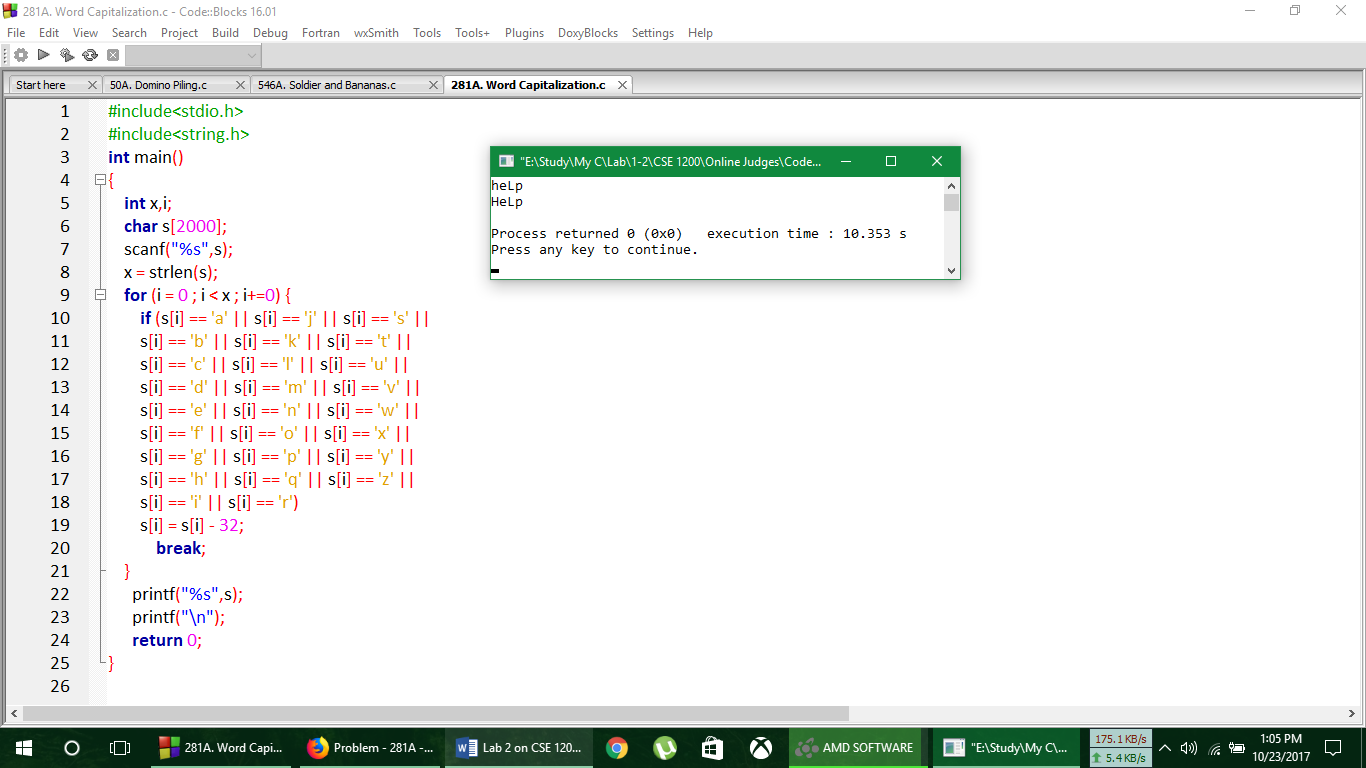
A single line contains a non-empty word. This word consists of lowercase and uppercase English letters. The length of the word will not exceed 103.

**Output**

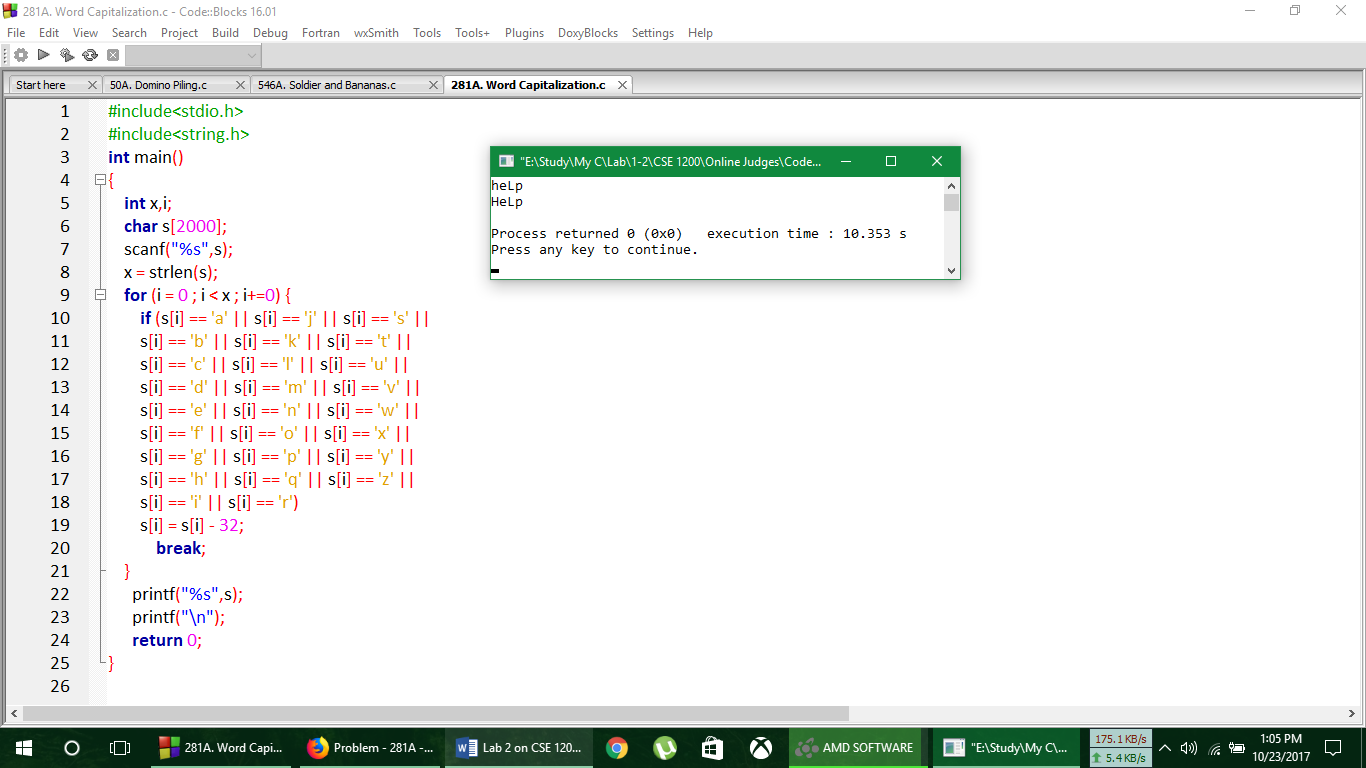
Output the given word after capitalization.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| ApPLe | ApPLe |
| konjac | Konjac |

**Code**

****

**Output**

****

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(x) .

**No. of Problem solved**: 19

**Codeforces Online Judge**

**Problem No. 266A**

**Stones on the Table**

There are *n* stones on the table in a row, each of them can be red, green or blue. Count the minimum number of stones to take from the table so that any two neighboring stones had different colors. Stones in a row are considered neighboring if there are no other stones between them.

**Input**

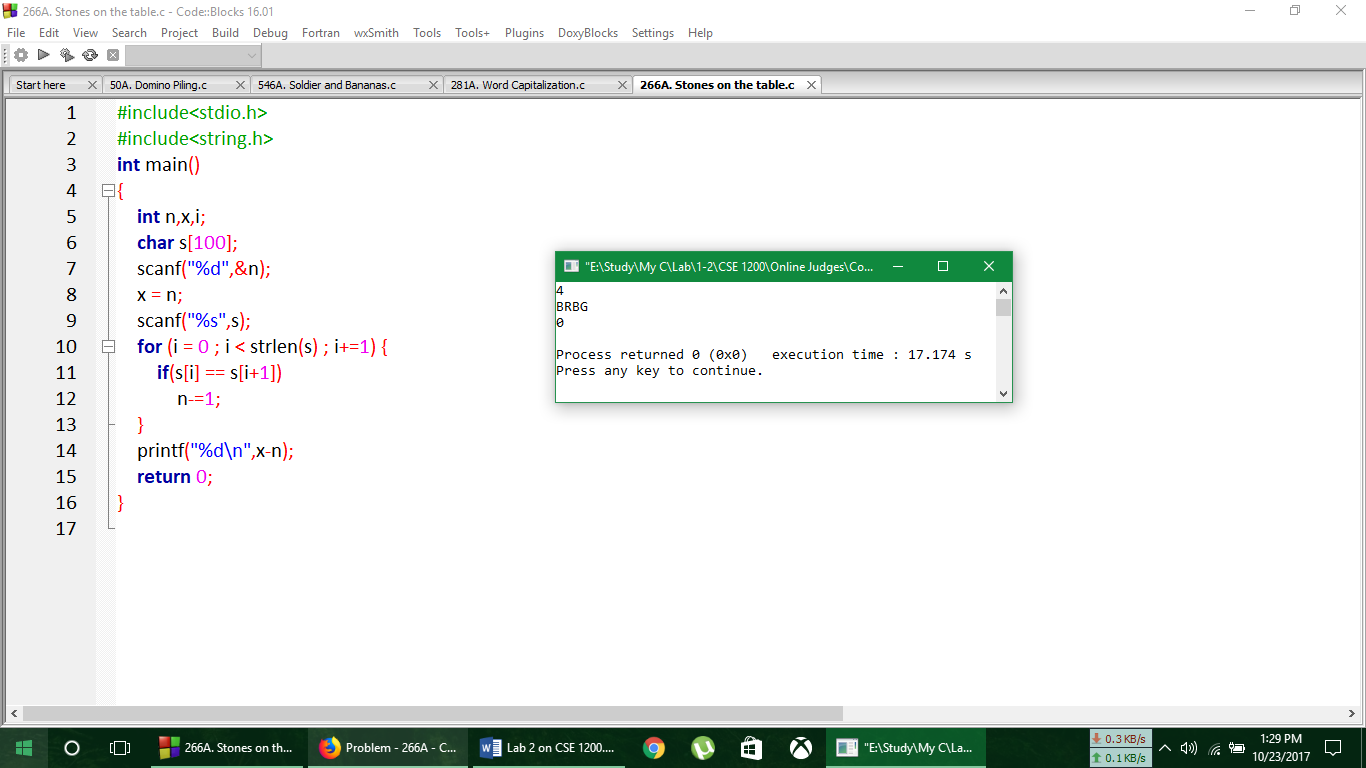
The first line contains integer *n* (1 ≤ *n* ≤ 50) — the number of stones on the table.

The next line contains string *s*, which represents the colors of the stones. We'll consider the stones in the row numbered from 1 to *n* from left to right. Then the *i*-th character *s* equals "R", if the *i*-th stone is red, "G", if it's green and "B", if it's blue.

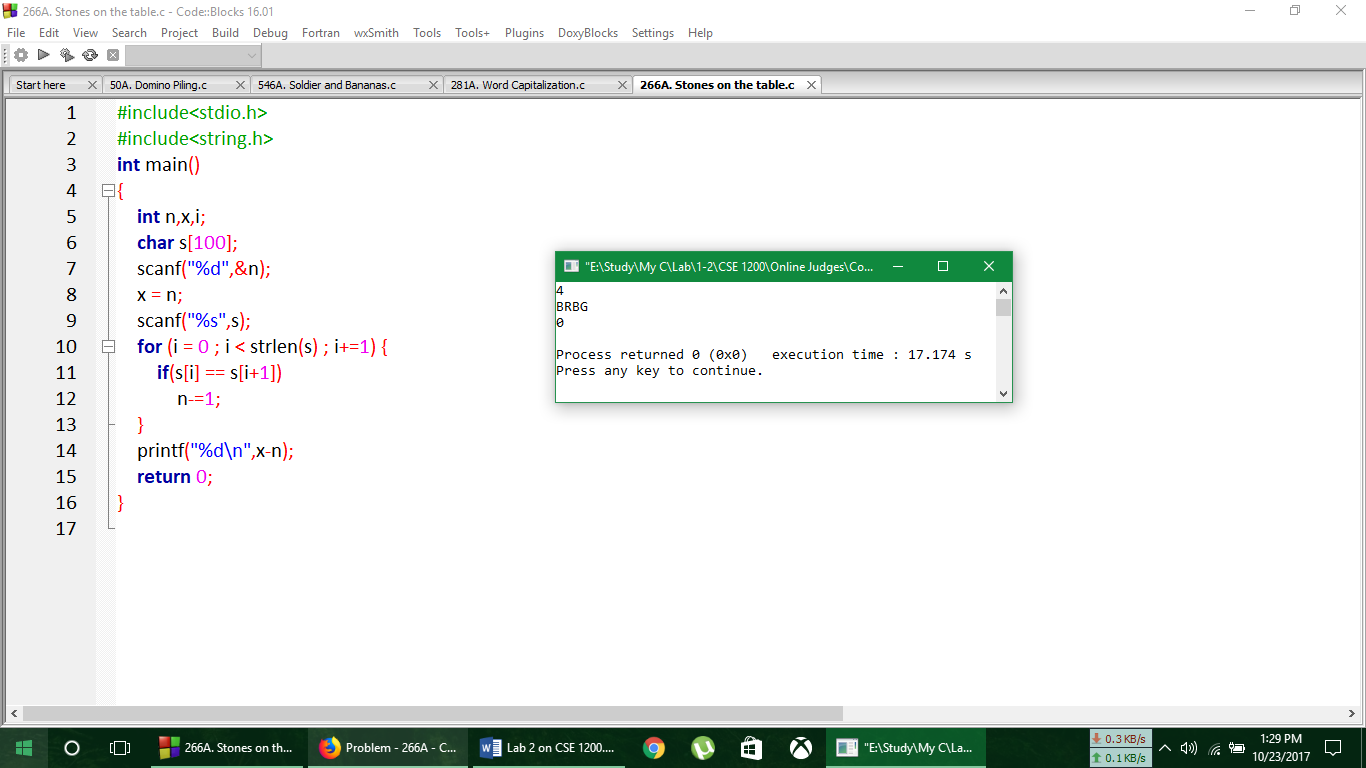
**Output**

Print a single integer — the answer to the problem.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 3 RRG | 1 |
| 4 BRBG | 0 |

**Code**

**Output**

****

**Verdict**

**Analysis of Solution**

Here the complexity of the above program is O(strlen(s)) .

**No. of Problem solved**: 20