# Problem 1 – Daily Calorie Intake

Kalinko is a junior software developer, who is mostly doing freelance work for random contractors. He has just been hired by a fitness instructor and is been tasked to create a software, which will assist the instructor in preparing healthy diets for his customers. In order to prepare the diets, the fitness instructor needs to know the necessary daily calorie intake for each of his clients. The daily calorie intake (DCI), is calculated by **multiplying the Basal Metabolic Rate (BMR) of the client, by a constant, which is determined by the number of workouts that the person does per week**. The BMR is calculated with the following formula:

**Men: BMR = 66.5 + (13.75 x weight in kg) + (5.003 x height in cm) – (6.755 x age in years)**

**Women: BMR = 655 + (9.563 x weight in kg) + (1.850 x height in cm) – (4.676 x age in years)**

Once the BMR is calculated, we can get the person's DCI, using the following table:

|  |  |
| --- | --- |
| **Number of workouts** | **Daily Calorie Intake** |
| No workouts | DCI = BMR \* 1.2 |
| 1–3 workouts per week | DCI = BMR \* 1.375 |
| 4–6 workouts per week | DCI = BMR \* 1.55 |
| 7–9 workouts per week | DCI = BMR \* 1.725 |
| Extra heavy workouts | DCI = BMR \* 1.9 |

Also, the fitness instructor that hired Kalinko, lives in the United States, which means that the weight and the height of his clients, will be given in an **Imperial format – pounds (lbs.) for the weight and inches for the height**. In order to make the BMR formulas work, Kalinko will have to **convert Imperial values into Metric values**. Assume that **1 inch has 2.54cm and 1kg has 2.2lbs**.

Your job is to help Kalinko with his first big contract and create the software for him. You will be given a person's weight in pounds (lbs.), height in inches, age, gender and number of weekly workouts, each at a separate line. Your only output, should be the person's daily calorie intake (DCI). The DCI should be **rounded down to the nearest integer number.**

### Input

The input should be read from the console. It will consist of exactly 5 input values, each at a separate line.

1. **W – weight in pounds (lbs.)**
2. **H – height in inches**
3. **A – age**
4. **G – gender**
5. **E – workouts per week**

### Output

* The output should be the calculated DCI. It should be a **single number, rounded down to the nearest integer number.**

### Constraints

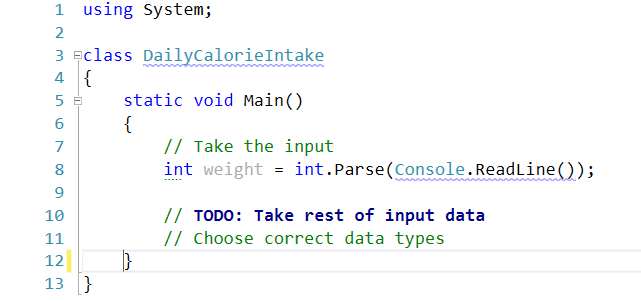
* The W, H, A and E inputs will be valid integers, in the range [-2,147,483,648 … 2,147,483,647]
* The G input will be a single character – 'm' for male or 'f' for female
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16MB.

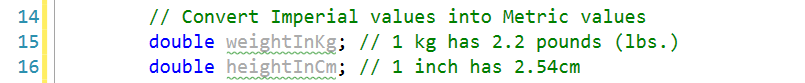
### Examples

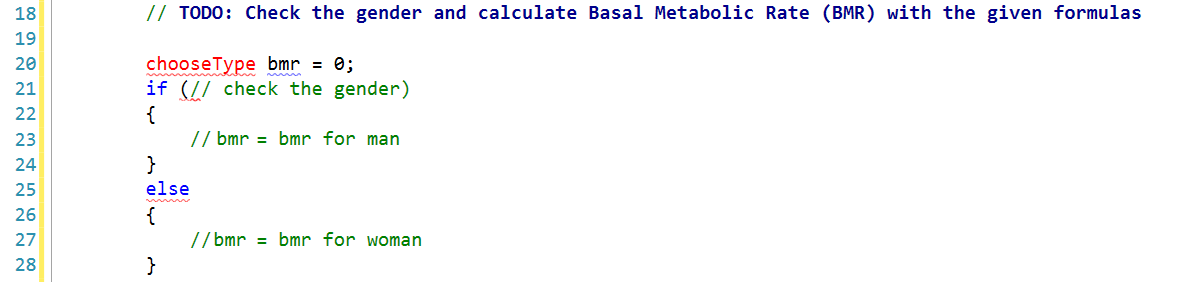
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Comments** | **Input** | **Output** | **Input** | **Output** |
| 154  70  27  m  0 | 2083 | 70 inches \* 2.54 = 177.8cm;  154lbs / 2.2 = 70kg;  66.5 + (13.75 \* 70) + (5.003 \* 177.8) – (6.755 \* 27) = 1736.1484 \* 1.2 = 2083.37808 | 130  63  21  f  7 | 2445 | 250  85  35  m  15 | 4698 |

## Step by step instructions

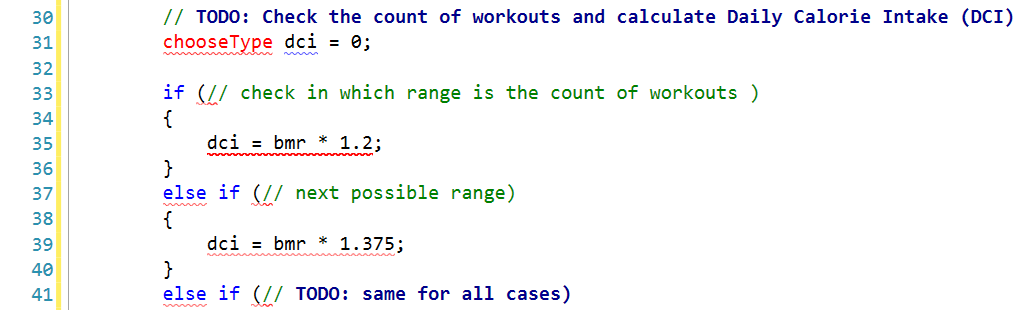
1. Let’s take the input data, carefully choose data types and if is needed parse them.



1. Next convert Imperial values into Metric values
2. Calculate Basal Metabolic Rate (**BMR**)



1. Let’s calculate daily calorie intake (DCI) which we need for the output



1. Last but not least print the result in console. Find the right way to round down to nearest integer.
2. Now you can submit your solution in [judge system](https://judge.softuni.bg/Contests/Practice/Index/104#0)

**Problem 2 – The football statistician**

Mr. Vulchan is a football statistician and his daily job is to watch football matches, collect the results and prepare a table of the league. Unfortunately, one day he fell off a cliff and hit his head badly. From that day he can’t remember any numbers and he needs some help with them. Your task is to prepare the league table for him**.** Since you are very good at programming, you decide to make a program to do this.

You should make a league table only for one of the leagues. It consists of 8 teams: Arsenal, Chelsea, Manchester City, Manchester United, Liverpool, Everton, Southampton and Tottenham.

You are given a string in the format **″team1** outcome **team2″** separated by **one or more whitespaces**. The **outcome** of the match will be one of the characters **[1, X, 2].** The character **′1′** represents a **win** for **team1**, the character **′2′** represents a **win** for **team2** and **′X′** represents **draw**.

* When **one of the teams** wins, it receives **3 points.** The **other team** receives **0 points**.
* In case of a **draw** the both teams receive **1 point**.

Mr. Vulchan will pay you **N** **euros** for every match. You should evaluate how much money you will obtain for this job and print it on the console in **leva**. Assume that **1** euro is **1.94lv**.

### Input

The input data should be read from the console.

* On the first line you will receive the payment for every match.
* On the next **N** lines you are given several input lines holding the match and its result. When you receive the command “**End of the league.**” the program should stop.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output data should be printed on the console.

* On the first line you should print the evaluated price for all matches in the league in **leva** rounded to **two** digits after the decimal point.
* On the next 8 lines you should print all the teams in **alphabetical order**, each on a separate line, together with the **points**, they have gained.

### Constraints

* The payment will be a floating point number in the range (-7.9 x 1028 to 7.9 x 1028) / (100 to 1028)
* Allowed working time for your program: 0.1 seconds.
* Allowed memory: 16 MB.
* Hint: The teams with more than one word are represented without **spacing** in **Pascal case.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 24.33  Chelsea 2 ManchesterCity  Everton 1 ManchesterUnited  Arsenal X Liverpool  Southampton 1 Tottenham  End of the league. | 188.80lv.  Arsenal - 1 points.  Chelsea - 0 points.  Everton - 3 points.  Liverpool - 1 points.  Manchester City - 3 points.  Manchester United - 0 points.  Southampton - 3 points.  Tottenham - 0 points. |

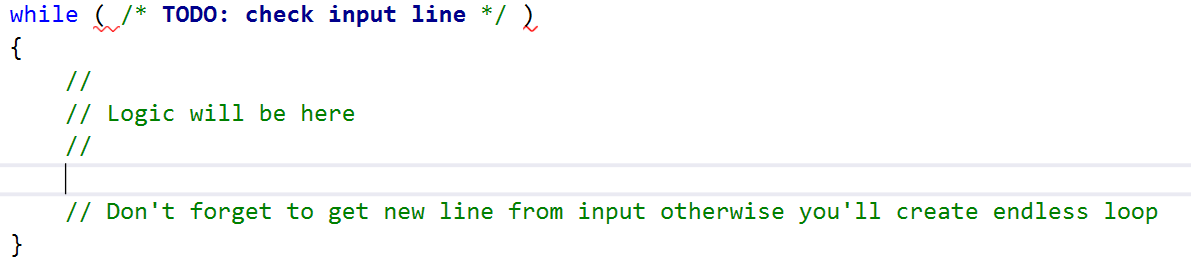
|  |  |
| --- | --- |
| **Input** | **Output** |
| 12.33  Chelsea 1 ManchesterUnited  Everton 1 ManchesterCity  Arsenal 1 Tottenham  Southampton 2 Liverpool  Southampton X ManchesterCity  Liverpool X Everton  Chelsea 2 Arsenal  ManchesterUnited X Tottenham  End of the league. | 191.36lv.  Arsenal - 6 points.  Chelsea - 3 points.  Everton - 4 points.  Liverpool - 4 points.  Manchester City - 1 points.  Manchester United - 1 points.  Southampton - 1 points.  Tottenham - 1 points. |

## Step By Step Instructions

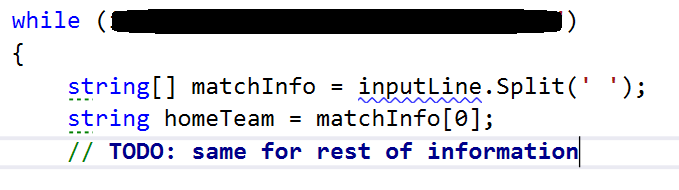
This problem has a lot of solutions. You can choose between switch/case (with or without extracted methods using out or ref) and some solution with array. For this lab we will switch.

1. First we should take the input
   * Take the payment for match – look constraints to find which data type you should use.
   * The input says: “On the next **N** lines you are given several input lines”. But you don’t know the value of **N**. “When you receive the command “**End of the league.**” the program should stop.”

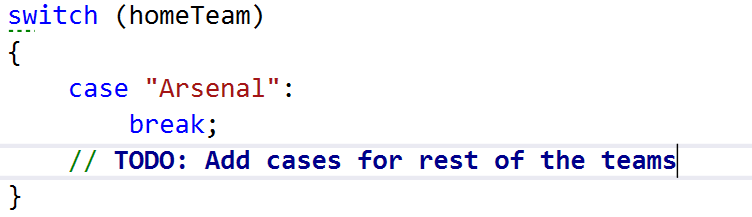
You should use while loop:



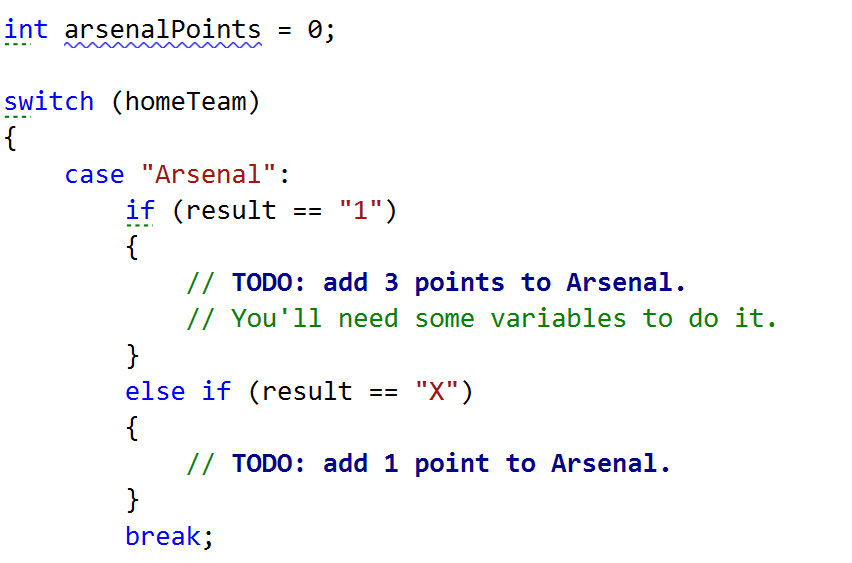
1. Now split the input line to get array with current info. And you can put values in variables.



1. Now after you have input information use it to solve the problem. As we say at the beginning we’ll use some switch to find which are the current teams. Let’s start with home team.

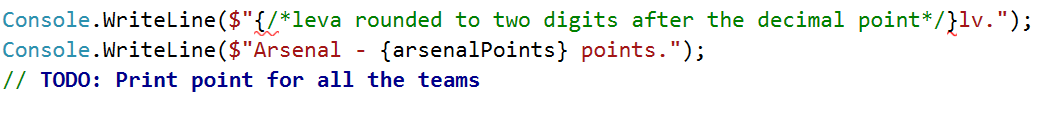


1. Now you should write what cases will do. Check the result if is “1” and add 3 point to the current team, and if is “X” add 1 point.



Do the same for all home teams. Teams points variables must be out of the while loop. Otherwise they will be zeroed оn each iteration.

1. Back to step 3 and do the same for **away teams**.
2. After you have all teams points let’s calculate the incomes.
   * First you should find count of matches. Create a variable witch is incremented in every iteration of the while loop.
   * The multiply the payment from input with the count of matches.
   * Don’t forget the amount of money you have now is in euro. You should convert them in leva. We assume that that one euro is 1.94 leva.
3. Last thing is to print the result on the console.
   * On the first line is leva rounded to two digits after the decimal point. Find way to do it.
   * Next 8 lines are the teams with their points.



1. Test the program with all examples
2. Copy all the code and test it in Judge – [https://judge.softuni.bg/Contests/Practice/Index/104#1](https://judge.softuni.bg/Contests/Practice/Index/104" \l "1)

**Problem 3 – Striped Towel**

The summer is almost over, but you still plan to have a summer vacation on the beach. Since our Black Sea is quite polluted nowadays, you plan to go to the Mediterranean Sea. A lot of people are going there, so you want to stand out of the crowd on the beach. You want to create your own design for a beach towel and have it in any possible size, so you can order to a very well-known local producer a full package for your family.

Your first steps in programming have led you to the conclusion that you can create a program to display your design. The **width** of the towel will be given as **integer** number. The **height** is calculated as the width, **multiplied by 1,5** and **rounded down** to the **nearest integer**. The design, you have chosen, is one with **diagonal stripes**. It follows this pattern:

The diagonal stripes are represented by **′#′.** The space between the stripes is represented as **′..′.** The towel always **starts** with a stripe in the **top-left corner** and follows the pattern **stripe-space-stripe-space**. The stripes are going **from top-right to bottom-left**. See the examples for more clarity.

### Input

The input should be read from the console. It will consist of a single line, holding the width of the towel.

### Output

The output should be the towel design, based on the input value.

### Constraints

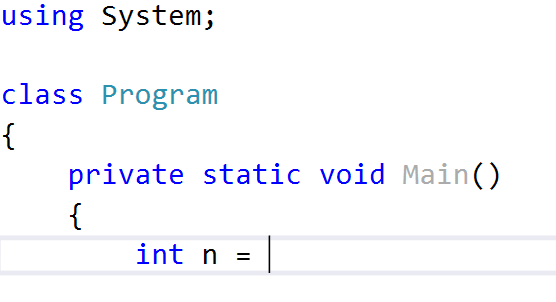
* The width will be valid **integer**, in the range [3 … 151].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 7 | #..#..#  ..#..#.  .#..#..  #..#..#  ..#..#.  .#..#..  #..#..#  ..#..#.  .#..#..  #..#..# |  | 9 | #..#..#..  ..#..#..#  .#..#..#.  #..#..#..  ..#..#..#  .#..#..#.  #..#..#..  ..#..#..#  .#..#..#.  #..#..#..  ..#..#..#  .#..#..#.  #..#..#.. |  | 5 | #..#.  ..#..  .#..#  #..#.  ..#..  .#..#  #..#. |

## Step By Step Instructions

1. First you should take the input number **“N”**

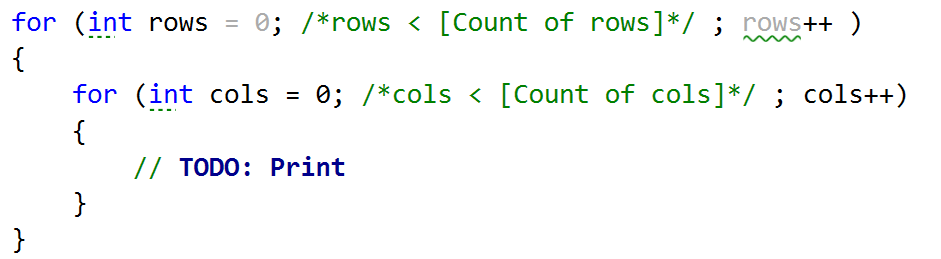


1. Then you should calculate the height of the towel. Which is the weight multiplied by 1.5.

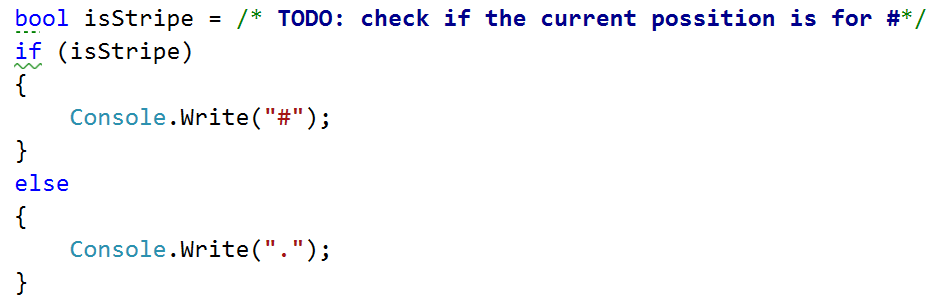


Ok, but you should round it down to the nearest integer. Hint you may cast it to integer or use some round methods.

1. You can imagine the towel as a matrix or table with rows and each symbol as a cell of row. You can print row by row. So you’ll need some loops. One for rows and one for cells of each row.



1. You should find when to print **‘.’** and when **‘#’**.Somehow you must find are you in right cell to print ‘#’.



1. If you find the way to print the dots and stripes you’ll see that you should print a new line. Think about when you should print it.
2. Test the program with all examples.
3. Copy all the code and test it in Judge – [https://judge.softuni.bg/Contests/Practice/Index/104#2](https://judge.softuni.bg/Contests/Practice/Index/104" \l "2)

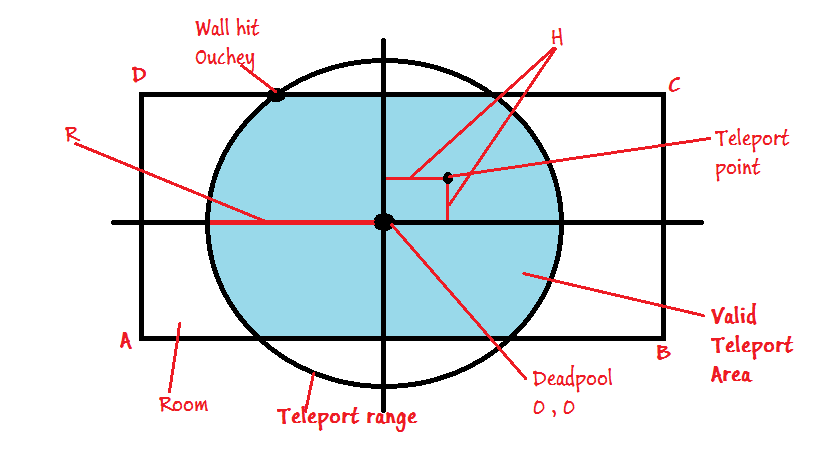
# Problem 4 – Teleport Points

Are you familiar with the Marvel universe? Chances are that you have at least heard of some of Marvel's most prominent heroes, such as Spider-Man, Captain America, Iron Man and Wolverine. Chances are, however that you haven't heard about some of Marvel's anti-heroes, mostly because anti-heroes tend to kill people, a LOT of people and thus don't usually get endorsed by the family-friendly mindset of Hollywood. Among these anti-heroes, is a character named Wade Wilson, also known as Deadpool, or “the merc with the mouth”. Wade is best known for his total disregard for human life, for being insane, for being practically immortal, and for having a love affair with Death herself. Deadpool's gear primarily consists of twin katana swords and two automated pistols. However, in some instances, he also possesses a teleportation device, which allows him to “body-slide” to short distances. The device has been destroyed during his last mission and he is now trying to acquire a new one. He already has the device built and he just needs someone to create the software for it, which will calculate possible teleportation points within a certain radius. Since you are the first computer programmer that he finds, you are appointed to do the job. You wouldn't want to disappoint Deadpool now, wouldn't you?

The program is very basic. The device can **teleport** a person **to any point** within a **radius R**. It also has a **step H**, which is the **distance between two neighboring points**. H has to be set manually, so **it cannot change** in the middle of a calculation. The device can only be used **within rectangular rooms** and the teleport **cannot pass through walls**. Deadpool knows that you are a junior developer, so your algorithm must work only **within a two-dimensional space**.

You will be given the coordinates of the **four points of the rectangle**, which will represent **the room**. You will also be given the **value R**, which is the **radius** of the device and the **value H**, which is the device's **step**. Deadpool's position will **always be 0,0**. Your task is to count **all possible points** Deadpool can **teleport** **to**. Points that will make him teleport **within a wall**, are **not considered valid**.

You should probably start working now. Deadpool does not regard patience as a virtue. He did draw you a pretty picture however, so that you can get a better understanding of your task.



### Input

The input should be read from the console and will consist of exactly 6 lines.

* The first **4 lines** will contain the **[X Y] coordinates** for each of the **four points** of the room.
* The **X** and the **Y** values will be separated by a **single space**.
* **Lines 5 and 6** will contain the values of the radius **R** and the step **H** respectfully.

1. **[X Y]** – coordinates for point A
2. **[X Y]** – coordinates for point B
3. **[X Y]** – coordinates for point C
4. **[X Y]** – coordinates for point D
5. **R** – Radius
6. **H** – Step

### Output

* The output should consist of a single number, representing the count of valid teleport points.

### Constraints

* **X and Y** will always be in the range [-30.5 … 30.5]
* **R and H** will always be in the range [0.1 … 30.5]
* **A, B, C, D** will always form a rectangle.
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16MB.

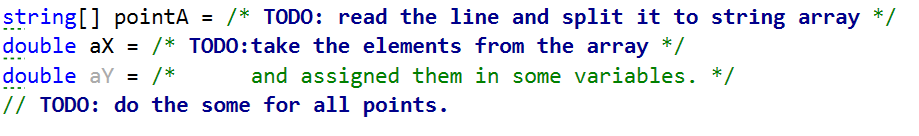
### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Explanation** |  | **Input** | **Output** |
| -20 -3  20 -3  20 3  -20 3  10  0.4 | 731 | The room is **40** wide and **6** tall. The circle has radius of **10**. Total points within the area with step **0.4** are **731**. |  | -7.2 -8.8  10.4 -8.8  10.4 9.1  -7.2 9.1  30.5  1 | 324 |

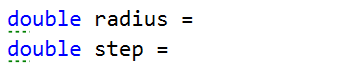
## Step by step instruction

* 1. Take the input
     + First let’s take the coordinates of rectangle’s edges.

Each of first 4 input lines will have the coordinates of the edges. You should split it and assigned them in some variables.



* + - Then the radius and step

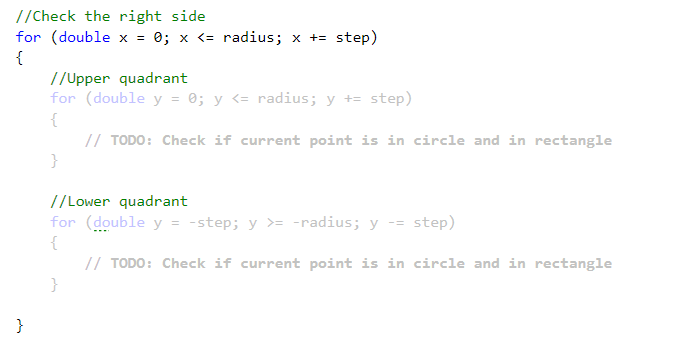


* 1. Because you have to count all possible movements let’s create some counter.

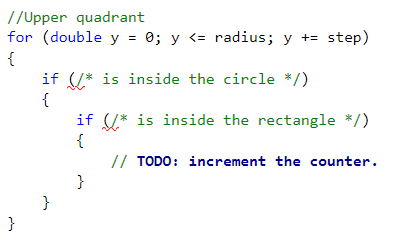


* 1. Let’s take the X and all possible Y for current X. You’ll need for loops. You can split the system in two parts: left and right. And each one of them to upper and lower.
* Let’s check the right side. You’ll need one loop iterates throw all possible X values from zero (start position) to radius.
* Inside this loop you’ll need another two for loops iterates throw all possible Y values from zero to radius(for upper quadrant) and for zero to minus radius(for lower quadrant)

The loop for lower quadrant starts from 0 – step. Y = 0 is already checked by previous loop.



* Now you iterate throw all possible positions, but you didn’t check if the point is in the circle and rectangle. Find in internet formula for point inside circle. Then check is it inside the rectangle.



* Do the same for lower quadrant.
  1. Get back to previous step (3) and do the same for left side. Be careful for limits of the loops. They are different.
  2. After you have the count of all possible positions, print it on the console
  3. Test the program with all examples.
  4. Copy all the code and test it in Judge – <https://judge.softuni.bg/Contests/Practice/Index/104#3>.

# Problem 5 - Wave Bits

Did you know that you can find waves in bits? They are sequences such as 1010101 (bits going up and down) which **always** start and end with 1. This is the binary representation of **1397221**. There are two wave sequences in it. The first one is nine bits long and the second – only three.

00000000000**101010101**000111100**101**

Your task is to find the longest of those sequences. After that, you have to delete it. In the current example, the resulting number would be

00000000000000000000000111100**101**

Its decimal representation is **485**. If there are **several wave sequences of equal length**, delete the **rightmo**st and do **not** touch the others!

Deleting the bits means removing them completely from the number, not setting them to zeroes! See the examples below to understand your task better.

Write a program that does the described operations and prints the resulting number on the console.

**Input**

On the only line of input, you will receive a single integer number.

The input will always be valid and in the format described, there is no need to check it explicitly.

**Output**

There is only line of output.

Print the decimal representation of the number after you delete the longest wave in it. If no such number is found, print “No waves found!”

**Constraints**

* The number will be an integer in range [0.. 18,446,744,073,709,551,615]
* There cannot be negative numbers in the input or output!
* Allowed working time: 0.1 seconds. Allowed memory: 16 MB.

**Examples**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comments |
| 66408933 | 127461 | 11111101010101000111100101 – We remove the wave bits and pad the number with a few zeroes to the left. The obtained number is  00000000011111000111100101 -> 127461 |

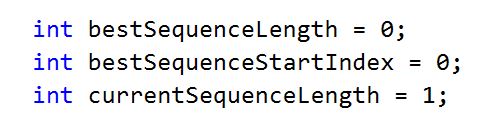
|  |  |  |
| --- | --- | --- |
| Input | Output | Comments |
| 67105255 | No waves found! | Binary representation of 67105255:  11111111111111000111100111. There are no waves in it. |

## Step By Step Instructions

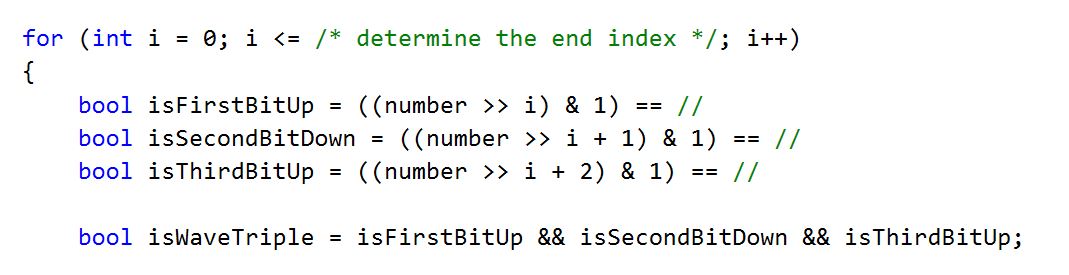
1. First off, check the **Input**, **Output** and **Constraints** sections. In the input we see that we receive only one line from the console. We need to parse it to it a number. There are a lot of different variable types for storing whole numbers in C# - short, int, uint, long … To determine the type we need, we have to check the **Constraints.** Notice that the range is given as [0... 18,446,744,073,709,551,615]. This means that we need to work with **ulongs.**

**C:\Users\Edu\Pictures\innput.JPG**

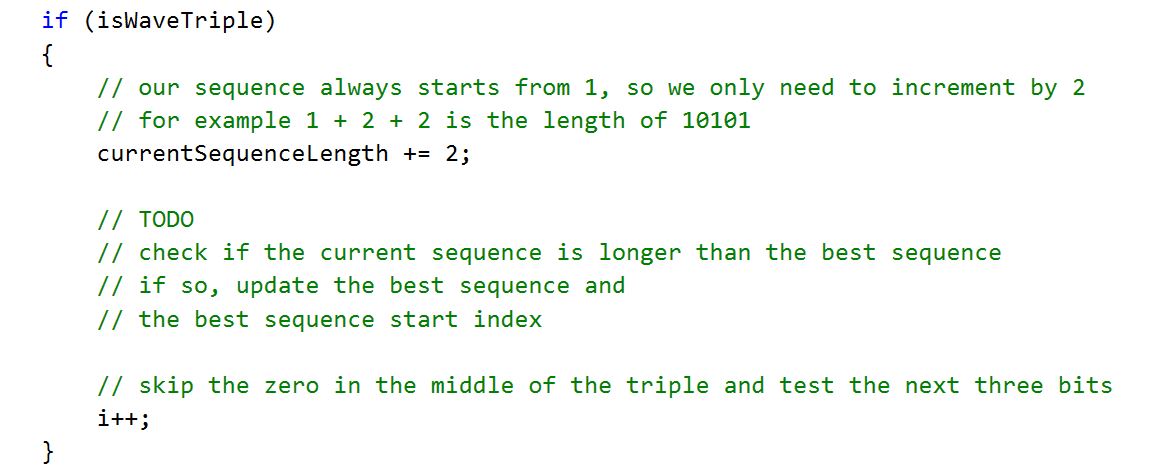
1. Next we will need several additional variables to store the wave sequence data. We need an integer that keeps the length of the current sequence we are testing, another which keeps the length of the best sequence, and a final one that stores the best sequence start index:



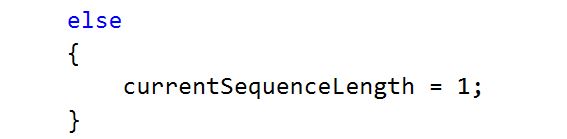
1. To determine the best sequence, we can look at its structure -> 10101010101. We can see that, actually it is a pattern 101 that is repeating itself. So we can search for those patterns and see if they are repeating. The idea is that we can create a **for loop** to traverse the bits of the number, and take the current bit, the next one, and the one after that. They are at positions **i, i + 1, i + 2.** The first should be 1, the second 0 and the third 1.



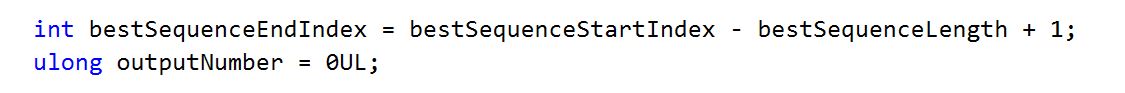
1. If first three Booleans are true, **isWaveTriple** will also be true.



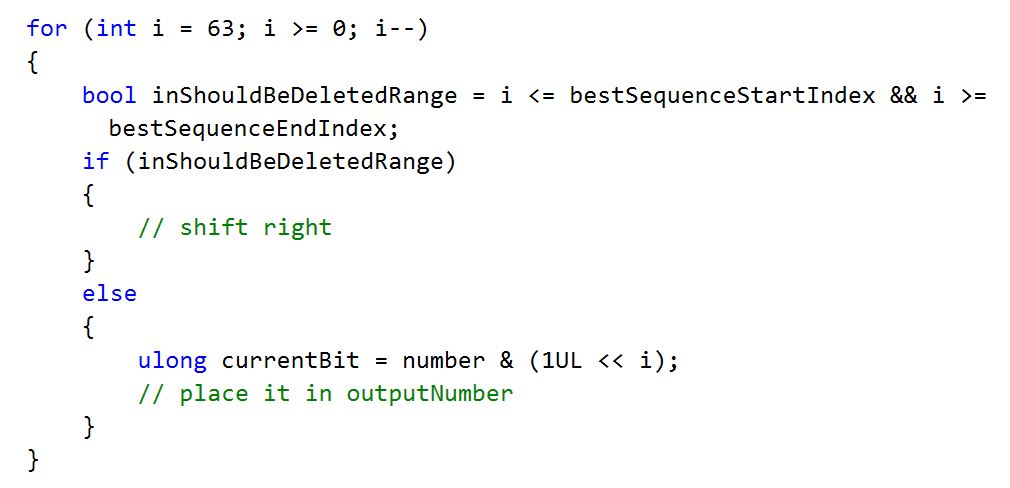
If it is not a wave sequence, we have to reset the counter, until we find a new one.



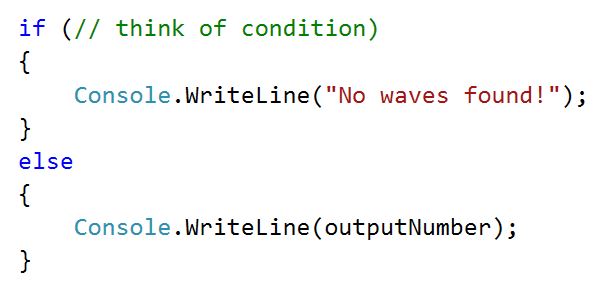
1. Once we have a best sequence, we will modify the number and delete from it the sequence. Doing it directly on the number is too difficult but we can copy the relevant bits in another number. We also need to know where the sequence ends. We have the length and the start index, so finding the end index is not too difficult.



1. Next we start modifying the output number. If the range of the bits is in the sequence range, we shift the number to the right. Else, we copy the bit from the initial number and place it at the same position of the output number.



1. The final task is to print the output. There are two cases we either have found a best sequence or not. Think of how you can determine that and print accordingly.



1. Test the program with all examples
2. Copy all the code and test it in Judge – <https://judge.softuni.bg/Contests/Practice/Index/104#4>