Exercise: Conditional Statements

Problems for exercise and homework for the "**Programming Basics**" course @ SoftUni Global.

Submit your solutions in the SoftUni Judge system at: https://judge.softuni.org/Contests/2390/Conditional-Statements-Exercise

1. Sum Seconds

Three athletes finish in a **matter of seconds** (between **1** and **50**). Write a program that reads the times of the competitors in seconds entered by the user and calculates their total time in the format "**minutes:seconds**". Display the seconds with **leading zero** (2 → "02", 7 → "07", 35 → "35").

Input	Output
35 45 44	2:04

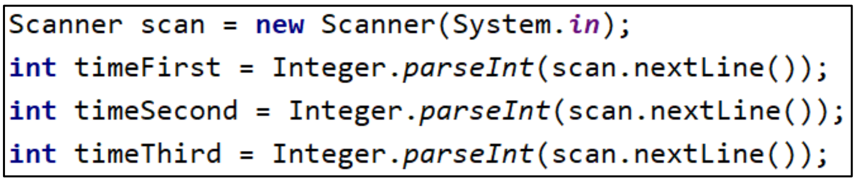
Input	Output
22 7 34	1:03

Input	Output
50 50 49	2:29

Input	Output
14 12 10	0:36

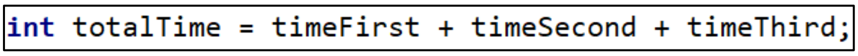
Hints and Guidelines

1. Read the input data (**competitors' seconds**):



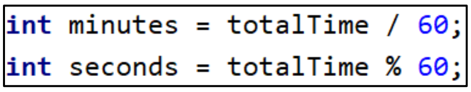
```java
Scanner scan = new Scanner(System.in);
int timeFirst = Integer.parseInt(scan.nextLine());
int timeSecond = Integer.parseInt(scan.nextLine());
int timeThird = Integer.parseInt(scan.nextLine());
```

2. Create a **new variable** to store the sum of the seconds of the **three competitors**:



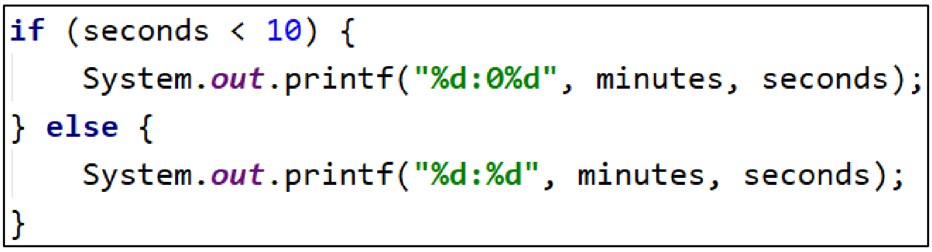
```java
int totalTime = timeFirst + timeSecond + timeThird;
```

3. Once you have found the **sum of the seconds**, you need to **convert them to minutes and seconds** (for example, if the sum is **85 seconds**, this is **1 minute and 25 seconds**, because **1 minute has 60 seconds**). Create two new variables. In the first variable, calculate **how many minutes the sum of seconds is by dividing the sum by 60**. In the second variable, calculate the seconds using division by the remainder (%). **Use division with remainder** (%) to take the remainder when dividing by 60, which is the remaining seconds. For example, you have a total of **134 seconds** (2 minutes and 14 seconds) after the integer division (/) of 60 we get 2 and the remainder 14, **which we take with the division by the remainder** (%).



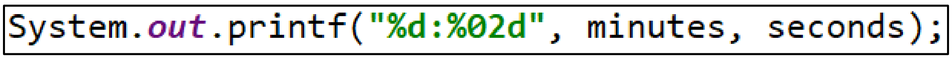
```java
int minutes = totalTime / 60;
int seconds = totalTime % 60;
```

4. Once you know **how many minutes and seconds** is the total, we need to print them in the correct format (**minutes:seconds**), and if the seconds are **less than 10** we need to **print 0 before the seconds**, otherwise we just print **the result in the given format**. To do this, check it with a **conditional statement** (if). You can use **`printf`** and **the integer template %d for printing**.



```java
if (seconds < 10) {
    System.out.printf("%d:0%d", minutes, seconds);
} else {
    System.out.printf("%d:%d", minutes, seconds);
}
```

*** Printing the result with a leading zero can also be done using the template %02d, through which we can show that we want our integer (seconds) to consist of two digits:**



```java
System.out.printf("%d:%02d", minutes, seconds);
```

2. Bonus Score

An **integer** is given - the starting number of points. **Bonus points** are added to it according to the rules described below. Write a program that calculates the **bonus points received by the number and the total number of points** (number + bonus).

- If the number is up to **100** inclusive, the bonus points are **5**.
- If the number is greater than **100**, the bonus points are **20%** of the number.
- If the number is greater than **1000**, the bonus points are **10%** of the number.

- Additional bonus points (added separately from the previous ones):
  - For an even number → + 1 point.
  - For a number ending in **5** → + 2 points.

Sample Input and Output

Input	Output
20	6.0 26.0

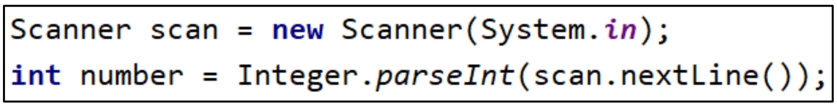
Input	Output
175	37.0 212.0

Input	Output
2703	270.3 2973.3

Input	Output
15875	1589.5 17464.5

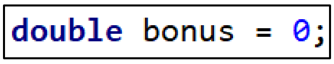
Hints and Guidelines:

1. Read the input data (**an integer**):



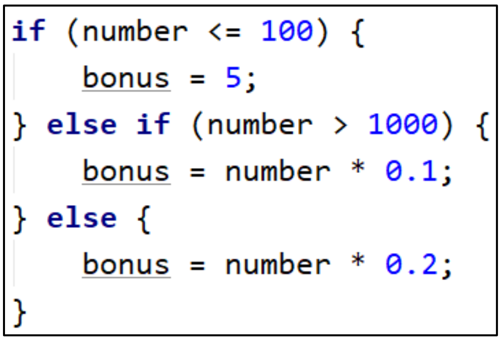
```java
Scanner scan = new Scanner(System.in);
int number = Integer.parseInt(scan.nextLine());
```

2. Create a **new variable of type double**, in which you will calculate the **accumulated bonus points**, giving it a starting value of 0.



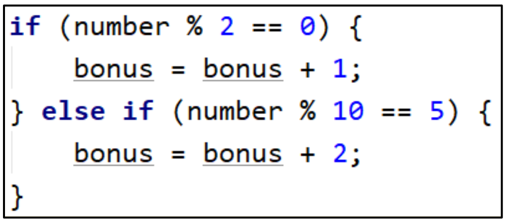
```java
double bonus = 0;
```

3. Make an **if-else-if construction** for **three conditions** to check the size of the number and calculate the bonus.



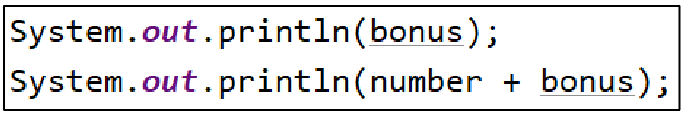
```java
if (number <= 100) {
    bonus = 5;
} else if (number > 1000) {
    bonus = number * 0.1;
} else {
    bonus = number * 0.2;
}
```

4. Create a new if-else-if construct to perform the conditions and **calculate the additional bonus**. If the **number is even, add 1 to the bonus accumulated** so far, and if it ends at 5, add 2 to the bonus. To check if a number is even, you must **divide it by 2**, and if you get a remainder divided by 0, then **the number is even**. but if you get a remainder of 1, it means that **the number is odd**. For example, the number 34 is even because 34/2 = 17 and the remainder is 0, and the number 35 is odd because 35/2 = 17 with a remainder of 1. To check if a number ends in 5 you have to divide the number by 10 and if get a remainder in division 5, so the number ends in 5. For example, the number 245/10 = 24 with the remainder 5.



```java
if (number % 2 == 0) {
    bonus = bonus + 1;
} else if (number % 10 == 5) {
    bonus = bonus + 2;
}
```

5. Print the results in **two lines**. On the first row is the **accumulated bonus** and on the second is the **final number**, which you will find by adding the **initial number** of points and **the bonus**.



```java
System.out.println(bonus);
System.out.println(number + bonus);
```

3. Time + 15 Minutes

Write a program that reads the hour and minutes of the 24-hour day entered by the user and calculates what time it will be in 15 minutes. Print the result in **hours:minutes**. The hours are always between 0 and 23, and the minutes are always between 0 and 59. The hours are written in one or two digits. Minutes are always displayed in two digits, with a leading zero when necessary.

Sample Input and Output

Input	Output
1 46	2:01

Input	Output
0 01	0:16

Input	Output
23 59	0:14

Input	Output
11 08	11:23

Input	Output
12 49	13:04

Sample Exam Problems

4. Toy Shop

Sophie has a toy store. She receives a large order that she must fulfill. With the money she will earn, she wants to go on a trip.

Toy prices:

- **Puzzle - 2.60 lv(BGN).**
- **Talking doll - 3 lv(BGN).**
- **Teddy bear - 4.10 lv(BGN).**
- **Minion - 8.20 lv(BGN).**
- **Truck - 2 lv(BGN).**

If the ordered toys are 50 or more, the store makes a 25% discount on the total price. Sophie must give 10% of the earned money for the rent of the store. Calculate whether the money will be enough for her to go on a trip.

Input Data

6 lines are read from the console:

1. **Price of the trip – floating-point in the interval [1.00 … 10000.00]**
2. **Number of puzzles – integer in the interval [0… 1000]**
3. **Number of talking dolls - integer in the interval [0 … 1000]**
4. **Number of teddy bears - integer in the interval [0 … 1000]**
5. **Number of minions - integer in the interval [0 … 1000]**
6. **Number of trucks - integer in the interval [0 … 1000]**

Output Data

On the console print:

- If the **money is enough** print:
  - "Yes! {**remaining money**} lv left."
- If the **money isn't enough** print:
  - "Not enough money! {**needed money**} lv needed."

The result must be formatted to the second decimal symbol.

Sample Input and Output

Input	Output	Comments
40.8 20 25 30 50 10	Yes! 418.20 lv left.	**Sum**: 20 * 2.60 + 25 * 3 + 30 * 4.10 + 50 * 8.20 + 10 * 2 = **680** BGN **Number of toys**: 20 + 25 + 30 + 50 + 10 = **135** **135 > 50 => 25% discount;** 25% from 680 = **170** BGN **discount** **Final price**: 680 – 170 = **510** BGN **Rent**: 10% from 510 BGN = **51** BGN **Profit**: 510 – 51 = **459** BGN **459 > 40.8** => 459 – 40.8 = **418.20** BGN **remain**
Input	**Output**	

320 8 2 5 5 1	Not enough money! 238.73 lv needed.	**Sum**: 8 * 2.60 + 2 * 3 + 5 * 4.10 + 5 * 8.20 + 1 * 2 = **90.3** BGN **Number of toys**: 8 + 2 + 5 + 5 + 1 = **21** **21 < 50 => no discount** **Rent**: 10% from 90.3 = **9.03** BGN **Profit**: 90.3 – 9.03 = **81.27** BGN **81.27 < 320** => 320 – 81.27 = **238.73** BGN are not enough

5. Godzilla vs. Kong

Filming for the long-awaited film "Godzilla vs. Kong" begins. Screenwriter Adam Wingard asks you to write a program to calculate **whether the funds provided are enough to shoot the film**. The photos **will require a certain number of extras**, **clothing** for each extra, and **decor**.

It is known that:

- The set for the film is worth **10% of the budget.**
- For more than **150 extras**, there is a **10% discount on clothing.**

Input Data

3 lines are read from the console:

- **Movie Budget- a floating-point number in the interval [1.00… 1000000.00]**
- **Number of extras - an inter in the range [1…500]**
- **Price for clothing of one extra – floating-point in the interval [1.00… 1000.00]**

Output Data

Two rows are printed on the console:

- If the money for decor and clothes **is more than the budget**:
  - "**Not enough money!**"
  - "**Wingard needs {needed money for the movie} leva more.**"
- If the money for decor and clothes **is less than or equal to the budget**:
  - "**Action!**"
  - "**Wingard starts filming with {money left} leva left.**"

The result must be **formatted** to the second decimal symbol.

Sample Input and Output

Input	Output	Comments
20000 120 55.5	Action! Wingard starts filming with 11340.00 leva left.	Sum for decor: 10% from 20000 = 2000 BGN Sum for clothes: 120 * 55.5 = 6660 BGN Total sum for the movie: 2000 + 6660 = 8660 BGN 20000 – 8660 = 11340 BGN remaining
15437.62 186 57.99	Action! Wingard starts filming with 4186.33 leva left.	Sum for decor: 10% from 15437.62 = 1543.762 BGN Sum for clothes: 186 * 57.99 = 10786.14 BGN Extras are more than 150 therefore there is 10% discount for clothes 10% from 10786.14 is 1078.614 10786.14 – 1078.614 = 9707.526 BGN for clothes Total sum for the movie: 1543.762 + 9707.526 = 11251.288 BGN

		15437.62 – 11251.288 = 4186.331 BGN left
9587.88 222 55.68	Not enough money! Wingard needs 2495.77 leva more.	Sum for decor: 10% from 9587.88 = 958.788 BGN Sum for clothes: 11124.864 BGN Total sum for the movie: 958.788 + 11124.864 = 12083.652 BGN 9587.88 – 12083.652 = 2495.77 BGN needed

6. World Swimming Record

Oliver decides to improve the World Record for long-distance swimming. On the console, we type: **the record that Oliver has to improve**, **the distance** in meters **he has to swim**, and **the time** in seconds **for which he swims a distance of 1 m**. To write a program that calculates whether he has done the record, it must be considered that: **the resistance of the water slows him down every 15 m by 12.5 seconds**. After calculating how many seconds Oliver will slow down, **the result should be rounded down to the nearest integer numbe**r.

Calculate the time in seconds for which Ivan will swim the distance and the difference from the World Record.

Input Data

3 lines are read from the console:

1. **The records in seconds – a floating-point number in the interval [0.00 … 100000.00]**
2. **The distance in meters – a floating-point number in the interval [0.00 … 100000.00]**
3. **The time in seconds for which he swims 1 meter - a floating-point number in the interval [0.00 … 1000.00]**

Output Data

Printing the console depends on the result:

- If **Oliver has improved the World Record (his time is less than the record)** we print:
  - **"Yes, he succeeded! The new world record is {time of Oliver} seconds."**
- If **he has NOT improved the record (his time is greater than or equal to the record)** we print:
  - **"No, he failed! He was {needed seconds} seconds slower."**

The result must be formatted to the second decimal symbol.

Sample Input and Output

Input	Output	Comments
10464 1500 20	No, he failed! He was 20786.00 seconds slower.	Oliver must swim 1500 meters: 1500 * 20 = 30000 seconds. On each 15 meters, we add 12.5 seconds to his time: 1500 / 15 = 100 * 12.5 = 1250 seconds. Total time: 30000 + 1250 = 31250 seconds. 10464 < 31250 The time he needed to improve the world record: 31250 - 10464 = 20786 seconds.
Input	**Output**	
55555.67 3017 5.03	Yes, he succeeded! The new world record is 17688.01 seconds.	Oliver must swim 3017 meters: 3017 * 5.03 = 15175.51 seconds. On each 15 meters, we add 12.5 seconds to his time: 3017/ 15 = 201 * 12.5 = 2512.50 seconds. Total time: 15175.51 + 2512.50 = 17688.01 seconds. Record is beaten: 55555.67 > 17688.01

7. Shopping

Peter wants to buy **N** video cards, **M** CPUs, and **P** number of RAM. If the number of video cards **is greater than** that of the processors, he receives a **15% discount** on the final bill. The following prices apply:

- Video card - **250 lv(BGN)** for one.
- CPU - **35%** from the total price of **purchased video cards**.
- RAM - **10%** from the total price of **purchased video cards**.

Calculate the **amount needed** to purchase the materials and calculate whether the **budget will be enough**.

Input Data

4 lines are read from the console:

- **Peter's budget – a floating-point number in the interval [0.0…100000.0]**
- **Number of video cards – an integer in the interval [0…100]**
- **Number of CPUs – an integer in the interval [0…100]**
- **Number of RAM – an integer in the interval [0…100]**

Output Data

On the console, print one row with the following text:

- If the budget is enough:
  `"You have` {budget left} `leva left!"`
- If the budget is not enough:
  `"Not enough money! You need` {needed sum} `leva more!"`

Format the result to the second decimal symbol.

Sample Input and Output

Input	Output	Comments
900 2 1 3	`You have 198.75 leva left!`	Budget: 900 BGN Sum for the requested video cards: 2 * 250 = 500 BGN Sum for a single CPU: 35% from 500 = 175 BGN Sum for requested CPUs: 1 * 175 = 175 BGN Sum for a single RAM: 10% from 500 = 50 BGN Sum for requested RAMs: 3 * 50 = 150 BGN Total sum: 500 + 175 + 150 = 825 BGN The number of video cards is greater than the number of CPUs, so he gets 15% discount from the final price: 825 – 15% = 701.25 BGN 701.25 <= 900 => the money is enough => money left: 900 – 701.25 = 198.75 BGN
920.45 3 1 1	`Not enough money! You need 3.92 leva more!`	Budget: 920.45 BGN Sum for the requested video cards: 3 * 250 = 750 BGN Sum for a single CPU: 35% from 750 = 262.50 BGN Sum for requested CPUs: 1 * 262.50 = 262.50 BGN Sum for a single RAM: 10% from 750 = 75 BGN Sum for requested RAMs: 1 * 75 = 75 BGN

		Total sum: 750 + 262.50 + 75 = 1087.50 BGN The number of video cards is greater than the number of CPUs, so he gets 15% discount from the final price: 1087.50 – 15% = 924.37 BGN 924.37 > 920.45 => the money isn’t enough => money needed 924.375 - 920.45 = 3.92 BGN

8. Lunch Break

During the lunch break, you want to watch an episode of your favorite series. Your task is to write a program that will find out if you have enough time to watch the episode. **During the holiday** you spend **time for lunch** and **time for rest**. Lunchtime will be **1/8 of the rest time**, and rest time will be **1/4 of the rest time**.

Input Data

3 lines are read from the console:

1. **Name of the series** - **a string**
2. **Episode duration** - **an integer in the range [10… 90]**
3. **Duration of the break** - **an integer in the range [10… 120]**

Output Data

On the console print:

- If you have **enough time to watch** the episode:
  "You have enough time to watch {name of series} and left with {time left} minutes free time."
- If you **don’t have enough time**:
  "You don't have enough time to watch {name of series}, you need {needed time} more minutes."

The time must be rounded to the nearest greater integer.

Sample Input and Output

Input	Output	Comments
Game of Thrones 60 96	You have enough time to watch Game of Thrones and left with 0 minutes free time.	Time for lunch: 96 * 1/8 = 12.0 Break time: 96 * 1/4 = 24.0 Remaining time: 96 - 12 - 24 = 60 The rest time is more or equal to the length of the episode, and we print the appropriate output.
Teen Wolf 48 60	You don't have enough time to watch Teen Wolf, you need 11 more minutes.	Time for lunch: 60 * 1/8 = 7.5 Break time: 60 * 1/4 = 15.0 Remaining time: 60 – 7.5 - 15 = 37.5 The rest time is less than the length of the episode, and we print the appropriate output.