**Softuniada 24 and 25 March 2018**

* **Awesome numbers**

Joro really likes numbers, but according to him some numbers are more awesome than others, help Joro write a program that can **classify numbers**. According to Joro numbers are classified according to **3 conditions**:

* Whether the number is **odd**
* Whether the number is **negative**
* Whether the **number can be divided by his favorite number without a remainder**

According to the 3 conditions, Joro classifies numbers in this way:

* If the number doesn’t **meet any of the conditions** (is not odd, is not negative and cannot be divided by his favorite number), then Joro thinks that number is “**boring”**
* If the number **meets only one condition**, then Joro thinks it is “**awesome**”
* If the number **meets 2 conditions**, then it is “**super awesome**”
* If the number **meets all 3 conditions**, then it is “**super special awesome**”

**Input**

Two lines are read from the console:

* On the first line, there will be **the number Joro wants to classify**
* On the second line, there will be Joro’s **favorite** **number**

**Output**

* On the only output line, print how Joro classifies the number:
* Does not meet conditions -> “**boring”**
* Meets 1 condition ->“**awesome**”
* Meets 2 conditions ->“**super awesome**”
* Meets 3 conditions ->“**super special awesome**”

**Constraints**

* **The number that Joro wants to classify will be an whole number in the interval [-1 000 000…1 000 000]**
* **Joro’s favorite number** will bea **whole number** in the interval **[-1 000 000…1 000 000]**
* Allowed time: **100 ms**. Allowed memory: **32 MB**.

**Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 13 7 | **awesome** | The number **13**: odd **√** negative **X** can be divided by **7** **X** since the number meets only 1 condition we print „**awesome**” |
| **Input** | **Output** | **Comments** |
| -27 9 | **super special awesome** | The number **-27**: odd **√** negative **√** can be divided by **9** **√** since the number meets all conditions, we print „**super special awesome**” |
| **Input** | **Output** | **Comments** |
| 151734 2152 | **boring** | The number **151734**: odd **X** negative **X** can be divided by **2152** **X** since the number doesn’t meet any of the conditions, we print „**boring**” |
| **Input** | **Output** | **Comments** |
| -1158 -6 | **super awesome** | The number **-1158**: odd **X** negative **√** can be divided by **-6** **√** since the number meets 2 of the conditions, we print „**super awesome**” |
| **Input** | **Output** | **Comments** |
| -33 12 | **super awesome** | The number **-33**: odd **√** negative **√** can be divided by **12** **X** since the number meets 2 of the conditions, we print „**super awesome**” |

* **Easter egg**

Write a program that reads from the console a **whole even number** **n**, as in the examples bellow. The Easter egg has **width** **– 5 \* n** and **height - 2 \* n + 3**. **Your task is to write a program that outputs an Easter egg**.

**Input**

* The input is read from the **console** and contains a single line – the number **n**.

**Output**

* Print on the console an **Easter egg** as in the examples.

**Constraints**

* The number **n** will always be a **whole even number** in the interval **[2…28]**
* Allowed time: **100 ms**. Allowed memory: **32 MB**.

**Examples**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 6 | ............\*\*\*\*\*\*............  ..........\*++++++++\*..........  ........\*\*++++++++++\*\*........  ......\*\*\*++++++++++++\*\*\*......  .....\*\*================\*\*.....  ....\*\*==================\*\*....  ...\*\*====================\*\*...  ...\*\*~~~~HAPPY EASTER~~~~\*\*...  ...\*\*====================\*\*...  ....\*\*==================\*\*....  .....\*\*================\*\*.....  ......\*\*\*++++++++++++\*\*\*......  ........\*\*++++++++++\*\*........  ..........\*++++++++\*..........  ............\*\*\*\*\*\*............ | 8 | ................\*\*\*\*\*\*\*\*................  ..............\*++++++++++\*..............  ............\*\*++++++++++++\*\*............  ..........\*\*\*++++++++++++++\*\*\*..........  ........\*\*\*\*++++++++++++++++\*\*\*\*........  .......\*\*======================\*\*.......  ......\*\*========================\*\*......  .....\*\*==========================\*\*.....  ....\*\*============================\*\*....  ....\*\*~~~~~~~~HAPPY EASTER~~~~~~~~\*\*....  ....\*\*============================\*\*....  .....\*\*==========================\*\*.....  ......\*\*========================\*\*......  .......\*\*======================\*\*.......  ........\*\*\*\*++++++++++++++++\*\*\*\*........  ..........\*\*\*++++++++++++++\*\*\*..........  ............\*\*++++++++++++\*\*............  ..............\*++++++++++\*..............  ................\*\*\*\*\*\*\*\*................ |

* **Bingo generator**

Misho and Deni thought up a new game called “Bingo”, but you have to help them and create a generator, which checks whether an entered **4-digit number** **wins**. The rules of the game are as follows:

* A 4 digit number is entered, from which we have to create two new **2-digit numbers**.
* The two **2-digit numbers** are created in the following way:
* **The first number** consists of the **thousands** digit and the **tens digit** in the **4 digit number**.
* **The second number** consists of the **hundreds digit** and the **units digit**.
* After we get the **two 2-digit numbers**, we need to calculate the **ceiling**. The **ceiling** is calculated by adding together the **two** **2-digit numbers.** For example **1234** – the first **2-digit number** is **13**, while the second is **24**, their sum is 13 + 24 = **37**, so the **ceiling** is equal to **37**.
* As a next step, we have to create a new 4-digit number – **the starting element**. **The starting element** is created by **appending the second 2-digit number to the back of the first 2-digit number**. For example, if the first 2-digit number is **13** and the second is **24**, the starting element will be **1324**.
* The next step is to generate all 4-digit numbers, which meet the following rules. The order of generation should be the following – we start from the **starting** **element**, first we **increase** **the** **second** **2-digit** **number** to its limit, after which we **increase** **the** **first** **2**-**digit** **number**, **reset** the second 2-digit number and **repeat** until there is no more viable numbers to generate. For example, with a **starting** **element** of **1123** and a **ceiling** of **37**, we generate them in the order 1123 -> 1124 -> 1125… 1137->1223->1224->1225…->1237->1323…:
* They should be **equal or bigger** than **the starting element**
* The **2-digit number** generated from the first 2 digits of the newly generated number, should be **less than or equa**l to the **ceiling**.
* The **2-digit** **number** generated from the last 2 digits of the newly generated number, should be **less than or equa**l to the **ceiling**.

If we take the number **1334** as an example, with a **ceiling** of **37** and a **starting** **element** **1324**, we can see that it meets the requirements**:**

* The number is bigger than the starting element => **1334 > 132**4
* The 2-digit number created from the first 2 digits is less than the ceiling => **13 < 37**
* The 2-digit number created from the last 2 digits is less than the ceiling => **34 < 37**

If we take the number **1342** as an example, it **is NOT valid**, because it does not meet the requirement that the second two digits should be less than the ceiling => **42 > 37**

* After creating the new **4-digit numbers** we need to check whether they **can be divided by 12 or 15** and to print them in two groups. If the amount of numbers that can be divided by 12 **is equal** to the amount of numbers that can be divided by 15, then the player who entered the 4-digit number is a winner.

**\*IMPORTANT: If any of the newly generated 4-digit numbers happens to be divisible by both 12 and 15 it should be printed in both groups.**

**Input**

* **A single line** – the entered 4-digit number

**Output**

**Three lines:**

* **First line :** **Dividing on 12:** {the newly generated 4-digit numbers that can be divided by 12 in ascending order}
* **Second line :** **Dividing on 15:** {the newly generated 4-digit numbers that can be divided by 15 in ascending order}
* **Third line:** If the amountof numbers divisibleby 12 is equal to the amount of numbers divisible by 15 print **“!!!BINGO!!!”**, alternatively print **“NO BINGO!”**

**Constraints**

* The 4-digit number will always be a whole number in the interval **[1000…9999]**
* If the ceiling is **bigger than or equal to 99**, then all **2-digit numbers** are valid.
* Allowed time: **100 ms**. Allowed memory: **32 MB**.

**Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| **1213** | **Dividing on 12: 1128 1224 1332 1428 1524 1632 1728 1824 1932 2028 2124 2232 2328 2424 2532 2628 2724 2832 2928 3024 3132 3228 3324 3432**  **Dividing on 15: 1125 1230 1425 1530 1725 1830 2025 2130 2325 2430 2625 2730 2925 3030 3225 3330**  **NO BINGO!** | We take the **first digit** (the thousands digit **1**) and the **third digit** (the tens digit **1**) from the 4-digit number => we get the **first** **2-digit number**, which is 11. After that we create the **other 2-digit number**, which is 23 (the hundreds digit **2** and the units digit **3**). The **maximum value** that any of the **two 2-digit numbers** can take is their sum => **11 + 23 = 34**. After we generate the numbers (1123, 1124, … , 1224, … , 3434), we check each of them and place it in one of the two groups – those divisible by 12 or those divisible by 15.  **Because the amount of numbers divisible by 12 (which is 24) is different than the amount of numbers divisible by 15 (which is 16) we print NO BINGO!** |
| **1214** | **Dividing on 12: 1128 1224 1332 1428 1524 1632 1728 1824 1932 2028 2124 2232 2328 2424 2532 2628 2724 2832 2928 3024 3132 3228 3324 3432 3528**  **Dividing on 15: 1125 1230 1335 1425 1530 1635 1725 1830 1935 2025 2130 2235 2325 2430 2535 2625 2730 2835 2925 3030 3135 3225 3330 3435 3525**  **!!!BINGO!!!** | |