Exercise: While Loop

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

Submit your solutions to the **SoftUni Judge** system at: https://judge.softuni.org/Contests/3696

1. Old Books

Sophie goes to her hometown after a very long period outside the country. On her way home, she sees her grandmother's old library and remembers her favorite book. Help Sophie by writing a program in which she enters the book (**string**) she is looking for. Currently, Sophie has not found her favorite book or does not check all the books in the library. The program must read on a new line the name of each subsequent book (**string**) that she checks. The books in the library are finished when you give the command "**No More Books**".

- **If she does not find the book:**
  - **`"The book you search is not here!"`**
  - **`"You checked {number of books} books."`**
- **If she finds the book print:**
  - **`"You checked {number of books} books and found it."`**

Sample Input and Output

Input	Output	Comments
Troy Stronger Life Style Troy	`You checked 2 books and found it.`	The book Sophie is looking for is Troy. First is Stronger, the second Life Style, and the third – Troy.
The Spot Hunger Games Harry Potter Torronto Spotify No More Books	`The book you search is not here!` `You checked 4 books.`	The book Sophie is looking for is "The Spot". The library contains 4 books. The first is Hunger Games, the second Harry Potter, the third Torronto, and the fourth Spotify. There are no more books and the searching ends. Sophie didn't find the book.
Bourne True Story Forever More Space The Girl Spaceship Strongest Profit Tripple Stella The Matrix Bourne	`You checked 10 books and found it.`	

Hints and Guidelines

1. Read the input from the console (**the name of the book you are looking for and the capacity of the library**):

```
String bookName = scan.nextLine();
```

2. Create **two new variables**. One will store **the number of books checked**. The other will be of the **boolean type**, giving it an initial value of **false** (**the book is not found**). In this variable, you will keep a value that will indicate whether **the book is found or not**. If the variable is **true** - the book is **found**, **otherwise** - the book **is not**.

```
int count = 0;
boolean isFound = false;
```

3. Create a new variable that will store information about the **current book you are checking**. Create a while loop in which you read one book each time until the book is found or until the library is full (**read "No More Books" command**). In the loop, check that the book you entered matches the one you are looking for, and if the check is correct, **change the value of the boolean variable** you created in the previous step to true (the book is found). Otherwise, increase by one the variable (**count**) you created in the second step.

```
String input = scan.nextLine();
while (!input.equals("No More Books")) {
    if (input.equals(bookName)) {
        isFound = true;
        break;
    }
    count++;
    input = scan.nextLine();
}
```

4. When the loop is completed, print the **two possible results**.

```
if (isFound) {
    System.out.printf("You checked %d books and found it.", count);
} else {
    System.out.printf("The book you search is not here!%nYou checked %d books.", count);
}
```

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#0

2. Exam Preparation

Write a program in which John solves exam **tasks until he receives** an "**Enough**" message from his lecturer. For each task solved, he receives a grade. The program must end by **reading the "Enough" command** or **if John receives the specified number of unsatisfactory grades**. Any grade **less than or equal to 4** is **unsatisfactory**.

Input Data

- **First row - number of unsatisfactory grades - integer in the range [1…5]**
- **Then two lines are read repeatedly:**
  - **Task name - text**
  - **Grade – integer in the range [2…6]**

Output Data

- If John reaches the "**Enough**" as a command, print in 3 lines:
  - **"Average score: {average grade}"**
  - **"Number of problems: {number of all problems}"**
  - **"Last problem: {name of last problem}"**

- If he receives the specified number of unsatisfactory grads:
  - **"You need a break, {number of unsatisfactory grades} poor grades."**

The average grade should be formatted to the second decimal place.

Sample Input and Output

Input	Output	Comments
3 Money 6 Story 4 Spring Time 5 Bus 6 Enough	Average score: 5.25 Number of problems: 4 Last problem: Bus	The number of allowed unsatisfactory grades is 3. The first problem is called Money, and his grade is 6. The second problem is called Story, and his grade is 4. The third problem is called Spring Time, and his grade is 5. The fourth problem is called Bus, and his grade is 6. The next command Enough, and the program stops. Average grade: 21 / 4 = 5.25 Number of solved problems: 4 Last problem: Bus
Input	**Output**	**Comments**
2 Income 3 Game Info 6 Best Player 4	You need a break, 2 poor grades.	The number of unsatisfactory evaluations is 2. The first problem is Income, and his grade is 3. The second problem is Game Info, and his grade is 6. The third problem is Best Player, and his grade is 4. John has reached the allowable number of unsatisfactory grades, and the program ends.

Hints and Guidelines

1. Read the input from the console:

```
int failedThreshold = Integer.parseInt(scan.nextLine());
```

2. Make four helper variables at the beginning:
   - **counter for unsatisfactory grades - with an initial value of 0**
   - **counter for the solved exercises - with an initial value of 0**
   - **the sum of all grades - with an initial value of 0**
   - **name of the last problem - with an initial value of blank text**
   - **whether the student failed or not**

```
int failedTimes = 0;
int solvedProblemsCount = 0;
double gradesSum = 0;
String lastProblem = "";
boolean isFailed = true;
```

3. Create a **while** loop that continues until the number of **unsatisfactory scores is less than the number you read** from the console. **Each** time you repeat the cycle, read the name of the problem and the grade for it.

```
while (failedTimes < failedThreshold) {
    String problemName = scan.nextLine();
    if ("Enough".equals(problemName)) {
        isFailed = false;
        break;
    }
}
```

4. If you receive the "**Enough**" command, change the value of **isfailed** to **true** and end the loop.
5. Each time you repeat the loop, **add his score to the sum of all his grades** and **increase the score counter**. If the score **is lower than or equal to 4**, increase the counter for **unsatisfactory grades**. Rewrite the name of the last problem.

```
while (failedTimes < failedThreshold) {
    String problemName = scan.nextLine();
    if ("Enough".equals(problemName)) {
        isFailed = false;
        break;
    }
    int grade = Integer.parseInt(scan.nextLine());
    if (grade <= 4) {
        failedTimes++;
    }
    gradesSum += grade;
    solvedProblemsCount++;
    lastProblem = problemName;
}
```

6. After the loop, if the **number of unsatisfactory grades** has reached the **maximum**, print the required message:

```
if (isFailed) {
    System.out.printf("You need a break, %d poor grades.", failedThreshold);
} else {
    System.out.printf("Average score: %.2f%n", gradesSum / solvedProblemsCount);
    System.out.printf("Number of problems: %d%n", solvedProblemsCount);
    System.out.printf("Last problem: %s", lastProblem);
}
```

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#1

3. Vacation

Jesse has decided to raise money for a trip and asks you to help her find out **if she can raise the necessary amount**. She saves or spends some of her money each day. If she wants to spend more than her cash, she will spend all of it and she **will have no money**.

Input Data

From the console read:

- **Money needed for the trip - a floating-point number in the range [1.00…25000.00]**
- **Cash - a floating-point number in the range [0.00...25000.00]**

Then two lines are read repeatedly:

- **Action type – text with "spend" and "save"**

- **The amount that you will save / spend – a floating-point in the range [0.01…25000.00]**

Output Data

The program must stop in the following cases:

- If **Jesse** only spends money **5 consecutive days**, print on the console:
  - `"You can't save the money."`
  - `"{total days passed}"`
- If **Jesse collects the money** for the holiday, print on the console:
  - `"You saved the money for {total days passed} days."`

Sample Input and Output

Input	Output	Comments
2000 1000 spend 1200 save 2000	You saved the money for 2 days.	Money needed for the trip: **2000** Cash available: 1000 spend – we subtract the next number from the money (1000 - 1200 = -200, which is less than 0 => available money = 0) ~ consecutive days spending money = 1 - total days: 1 save – we add the next number to the money (0 + 2000 = 2000) ~ consecutive days spending money = 0 - total days: 2 Cash available (2000) >= Money needed for the trip (**2000**)
110 60 spend 10 spend 10 spend 10 spend 10 spend 10	You can't save the money. 5	Money needed for the trip: **110** Cash available: **60** spend - we subtract the next number from the money (60 - 10 = 50) ~ consecutive days spending money = 1 - total days: 1 spend - we subtract the next number from the money (50 - 10 = 40) ~ consecutive days spending money = 2 - total days: 2 spend - we subtract the next number from the money (40 - 10 = 30) ~ consecutive days spending money = 3 - total days: 3 spend - we subtract the next number from the money (30 - 10 = 20) ~ consecutive days spending money = 4 - total days: 4 spend - we subtract the next number from the money (20 - 10 = 10) ~ consecutive days spending money = 5 - total days: 5 5 consecutive days spends money => cash available: 10 Cash available (10) < Money needed for the trip (**110**)
250 150 spend 50 spend 50 save 100	You saved the money for 4 days.	Money needed for the trip: **250** Cash available: 150 spend - we subtract the next number from the money (150 - 50 = 100) ~ consecutive days spending money = 1 - total days: 1 spend - we subtract the next number from the money (100 - 50 = 50)

save 100		~ consecutive days spending money = 2 - total days: 2 save - we subtract the next number from the money (50 + 100 = 150) ~ consecutive days spending money = 0 - total days: 3 save - we subtract the next number from the money (150 + 100 = 250) ~ consecutive days spending money = 0 - total days: 4 Cash available (250) >= Money needed for the trip (**250**)

Hints and Guidelines

1. Read input from the console:

```
double neededMoney = Double.parseDouble(scan.nextLine());
double ownedMoney = Double.parseDouble(scan.nextLine());
```

2. Add **two helping variables** at the beginning to track **the number of days elapsed and the number of consecutive days Jesse spends money**. Initialize both variables with a **value of zero:**

```
int daysCounter = 0;
int spendingCounter = 0;
```

   Create a **while loop** that repeats as long as Jesse's money is **less than the money she needs** for the trip, and the consecutive day counter is **less than 5**. Each time you repeat the loop, **read two lines** from the console - the first line is text - **spend** or **save**, and the second - the money that Jesse saved or spent. Also increase the day counter by 1:

```
while (ownedMoney < neededMoney && spendingCounter < 5) {
    String command = scan.nextLine();
    double money = Double.parseDouble(scan.nextLine());
    daysCounter++;
}
```

3. Check if Jesse **spends** or **saves** for the day:
   - if she **saves**, add the saved money to hers and **reset the counter for consecutive days**;
   - if she spends, subtract from her money the amount she spent and **increase the counter for the consecutive days** she spends. Check if Jesse's money has **become less than zero** and if so, she has run out of money

```
if ("save".equals(command)) {
    ownedMoney += money;
    spendingCounter = 0;
} else if ("spend".equals(command)) {
    ownedMoney -= money;
    spendingCounter += 1;
    if (ownedMoney < 0) {
        ownedMoney = 0;
    }
}
```

4. After the loop, check that Jesse has spent money on **five consecutive days** and print the message. Also check that Jesse **has saved the money** and, if successful, print the message:

```
if (spendingCounter == 5) {
    System.out.println("You can't save the money.");
    System.out.println(daysCounter);
}
if (ownedMoney >= neededMoney) {
    System.out.printf("You saved the money for %d days.", daysCounter);
}
```

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#2

4. Walking

Sophie wants to start a healthy lifestyle and has set herself the goal of walking 10,000 steps every day. However, some days she is very tired from work, and she will want to go home before she achieves her goal. Write a program that **reads from the console how many steps** it takes each time she goes out after the day and when it achieves its goal, print **`"The goal is achieved! Good job!"`** and how many more steps she took **`"{the difference between the steps} steps above the goal!"`**.

If she wants to **go home before that**, she will enter the **`"Going home"`** command and **enter the steps she took while returning home**. Then, if it fails to achieve its goal, print on the console: **`"{step difference} more steps to reach goal."`**

Sample Input and Output

Input	Output	Input	Output
1000 1500 2000 6500	Goal reached! Good job! 1000 steps over the goal!	1500 300 2500 3000 Going home 200	2500 more steps to reach goal.
Input	**Output**	**Input**	**Output**
1500 3000 250 1548 2000 Going home 2000	Goal reached! Good job! 298 steps over the goal!	125 250 4000 30 2678 4682	Goal reached! Good job! 1765 steps over the goal!

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#3

5. Coins

Vending machine manufacturers wanted to make their machines return **as few change coins as possible**. Write a program that receives **an amount** - **the money** that needs to be returned and calculates **with how few coins this can be done**.

Coins: 2.00 EUR, 1.00 EUR, 0.50 EUR, 0.20 EUR, 0.10 EUR, 0.05 EUR, 0.02 EUR, 0.01 EUR

Sample Input and Output

Input	Output	Comments
1.23	4	Our change is 1,23 EUR. The machine returns it to us with 4 coins: a coin of 1 EUR, a coin of 0.20 EUR, a coin of 0.02 EUR, and a coin of 0.01 EUR.
2	1	Our change is 2.00 EUR. The machine returns it to us with 1 coin of 2 EUR.
0.56	3	Our change is 0.56 EUR. The machine returns it to us with 3 coins: a coin of 0.50 EUR, a coin of 0.05 EUR and a coin of 0.01 EUR.
2.73	5	Our change is 2.73 EUR. The machine returns it to us with 5 coins: a coin of 2 EUR, a coin of 0.50 EUR, a coin of 0.20 EUR, a coin of 0.02 EUR and a coin of 0.01 EUR.

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#4

6. Cake

You are invited to the 30th birthday of your friend, and he has bought a huge cake. However, he does not **know how many pieces his guests can take from it.** Your task is to write a program that calculates the number of pieces that the cake must be split to. You will get the **dimensions of the cake in centimeters** (width and length - **integers** in the range [1 ... 1000]) and then on each line, until you receive the command "**STOP**" or until the cake is finished, the **number of pieces that guests take from** it. The pieces are square with a size of **1 cm**.

On the console print:

- **"{number of pieces} pieces are left."** – if you receive a **"STOP"** and there are pieces left.
- **"No more cake left! You need {number of missing pieces} pieces more."**

Sample Input and Output

Input	Output	Comments
10 10 20 20 20 20 21	No more cake left! You need 1 pieces more.	The cake is long **10cm** and has width **10cm** => Number of piece = **10** * **10** = **100** 1st take -> 100 - 20 = 80 2nd take -> 80 - 20 = 60 3rd take -> 60 - 20 = 40 4th take -> 40 - 20 = 20 5th take -> 20 - 21 = -1 < 0 => no more pieces left, 1 piece is not enough
10 2 2 4 6 STOP	8 pieces are left.	The cake is long 10cm and has width **2**cm => Number of piece = 10 * 2 = 20 1st take -> 20 - 2 = 18 2nd take -> 18 - 4 = 14 3rd take -> 14 - 6 = 8 4th take -> command STOP =>pieces left: 8

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#5

7. Moving

On Jose's eighteenth birthday, he decided to move out. He packed his luggage in boxes and found a suitable advertisement for an apartment for rent. He starts carrying his luggage in parts because he can't carry it all at once. He has limited free space in his new home, where he can place his belongings so that the place is suitable for living. Write a program that **calculates the free volume of Jose's home that remains after moving his luggage**.

Note: One box has dimensions: 1m. x 1m. x 1m.

Input Data

The user enters the following data on separate lines:

1. **Width of free space – an integer in the range [1...1000]**
2. **Length of free space – an integer in the range [1...1000]**
3. **Height of free space – an integer in the range [1...1000]**
4. **On the next lines (until receiving the command "Done") –the number of boxes that are moved to the apartment – an integer in the range [1...10000]**

The program should finish by reading the "Done" command or if the free space runs out.

Output Data

On the console print:

- If you receive **"Done"** and there is free space:

  "{number of free cubic meters left} Cubic meters left."

- If there is no free space, before typing **"Done"** print:

  "No more free space! You need {number of missing cubic meters} Cubic meters more."

Sample Input and Output

Input	Output	Comments
10 10 2 20 20 20 20 122	No more free space! You need 2 Cubic meters more.	10 * 10 * 2 = 200 cubic meters available 20 + 20 + 20 + 20 + 122 = 202 cubic meters 200 - 202 = 2 needed cubic meters
10 1 2 4 6 **Done**	10 Cubic meters left.	10 * 1 * 2 = 20 cubic meters available 4 + 6 = 10 cubic meters 20 - 10 = 10 cubic meters

Testing in the Judge System

Test the solution to this problem here: https://judge.softuni.org/Contests/Compete/Index/3696#6