

Lab: Linear Data Structures

This document defines the exercises for ["Java Advanced" course @ Software University](#). Please submit your solutions (source code) of all below described problems in [Judge](#).

I. Arrays and Lists

1. Encrypt, Sort and Print Array

Write a program that reads a **sequence of strings** from the console. Encrypt every string by summing:

- The code of **each vowel multiplied by the string length**
- The code of **each consonant divided by the string length**

Sort the **number** sequence in ascending order and print it on the console.

On first line, you will always receive the number of strings you have to read.

Examples

Input	Output	Comments
4 Peter Maria Katya Todor	1032 1071 1168 1532	Peter = 1071 Maria = 1532 Katya = 1032 Todor = 1168
3 Sofia London Washington	1396 1601 3202	Sofia = 1601 London = 1396 Washington = 3202

Hints

- Thinks about the **Arrays** class
- You might help yourself with the **code** below:

```
int n = Integer.parseInt(scanner.nextLine());
String[] names = new String[n];
for (int i = 0; i < n; i++) {
    names[i] = scanner.nextLine();
}
```

2. Split by Word Casing

Read a **text**, split it into words and distribute them into **3 lists**.

- **Lower-case words** like "programming", "at" and "databases" – consist of lowercase letters only.
- **Upper-case words** like "PHP", "JS" and "SQL" – consist of uppercase letters only.
- **Mixed-case words** like "C#", "SoftUni" and "Java" – all others.

Use the following **separators** between the words: , ; : . ! () " ' \ / [] **space**

Print the 3 lists as shown in the example below.

Examples

Input	Output
Learn programming at SoftUni: Java, PHP, JS, HTML 5, CSS, Web, C#, SQL, databases, AJAX, etc.	Lower-case: programming, at, databases, etc Mixed-case: Learn, SoftUni, Java, 5, Web, C# Upper-case: PHP, JS, HTML, CSS, SQL, AJAX

Hints

- **Split** the input text using the above described **separators**.
- **Process** the obtained **list of words** one by one.
- Create 3 lists of words (initially empty): lowercase words, mixed-case words and uppercase words.
- Check each word and append it to one of the above 3 lists:
 - Count the **lowercase letters** and **uppercase letters**.
 - If all letters are **lowercase**, append the word to the lowercase list.
 - If all letters are **uppercase**, append the word to the uppercase list.
 - Otherwise the word is considered mixed-case → append it to the mixed-case list.
- Print the obtained 3 lists as shown in the example above.

II. Multidimensional Arrays

3. Sum Matrix Elements

Write a program that **reads a matrix** from the console and prints:

- The count of **rows**
- The count of **columns**
- The sum of all **matrix's elements**

On the first line you will get the dimensions of the matrix in format **{rows, columns}**. On the next lines you will get the elements for each **row** separated with a coma.

Examples

Input	Output
3, 6 7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0	3 6 76

Hints

- Help yourself with the code below for reading the matrix
- Try to use a **foreach**-loop

```

for (int row = 0; row < matrix.length; row++) {
    String[] reminder = scanner.nextLine().split( regex: ", ");
    for (int col = 0; col < matrix[0].length; col++) {
        matrix[row][col] = Integer.parseInt(reminder[col]);
    }
}

```

4. Maximum Sum of 2x2 Submatrix

Write a program that **reads a matrix** from the console. Then find the biggest sum of a **2x2 submatrix**. Print the submatrix and its sum.

On the first line you will get the dimensions of the matrix in format **{rows, columns}**. On the next lines you will get the elements for each **row** separated with a coma.

Examples

Input	Output
3, 6 7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0	9 8 7 9 33
2, 4 10, 11, 12, 13 14, 15, 16, 17	12 13 16 17 58

Hints

- Ensure that your program doesn't throw an `IndexOutOfBoundsException()`

III. Working with Stacks

5. Simple Calculator

Create a **simple calculator** that can **evaluate simple expressions** that will not hold any operator different from addition and subtraction. There will not be parentheses or operator precedence.

Solve the problem **using a Stack**.

Examples

Input	Output
2 + 5 + 10 - 2 - 1	14
2 - 2 + 5	5

Hints

- Use an `ArrayDeque<>`

- Consider using the **add()** method
- You can either
 - add the elements and then pop them out
 - or push them and reverse the stack

6. Decimal to Binary Converter

Create a simple program that **can convert a decimal number to its binary representation**. Implement an elegant solution **using a Stack**.

Print the binary representation back at the terminal.

Examples

Input	Output
10	1010
1024	10000000000

Hints

- If the given number is 0, just print 0
- Else, while the number is greater than zero, divide it by 2 and push the remainder into the stack

```
while (decimal != 0) {
    stack.push(decimal % 2);
    decimal /= 2;
}
```

- When you are done dividing, pop all reminders from the stack, that is the binary representation

7. Matching Brackets

We are given an arithmetical expression with brackets. Scan through the string and extract each sub-expression.

Print the result back at the terminal.

Examples

Input	Output
1 + (2 - (2 + 3) * 4 / (3 + 1)) * 5	(2 + 3) (3 + 1) (2 - (2 + 3) * 4 / (3 + 1))
(2 + 3) - (2 + 3)	(2 + 3) (2 + 3)

Hints

- Use a stack, namely an **ArrayDeque()**
- Scan through the expression searching for brackets
 - If you find an opening bracket, push the index into the stack

- If you find a closing bracket pop the topmost element from the stack. This is the index of the opening bracket.
- Use the current and the popped index to extract the sub-expression

```
if (ch == '(') {
    stack.push(index);
}
else if (ch == ')') {
    int startIndex = stack.pop();
    String contents = expression.substring(startIndex, index + 1);
    System.out.println(contents);
}
```

IV. Working with Queues

8. Hot Potato

Hot potato is a game in which **children form a circle and start passing a hot potato**. The counting starts with the first kid. **Every n^{th} toss the child left with the potato leaves the game**. When a kid leaves the game, it passes the potato forward. This continues repeating **until there is only one kid left**.

Create a program that simulates the game of Hot Potato. **Print every kid that is removed from the circle**. In the end, **print the kid that is left last**.

Examples

Input	Output
Mimi Pepi Toshko 2	Removed Pepi Removed Mimi Last is Toshko
Gosho Pesho Misho Stefan Krasi 10	Removed Krasi Removed Pesho Removed Misho Removed Gosho Last is Stefan
Gosho Pesho Misho Stefan Krasi 1	Removed Gosho Removed Pesho Removed Misho Removed Stefan Last is Krasi

9. Math Potato

Rework the previous problem so that a **child is removed only on a prime cycle** (cycles start from 1)

If a **cycle is not prime**, just **print the child's name**.

As before, print the name of the child that is left last.

Examples

Input	Output
Mimi Pepi Toshko	Removed Pepi

2	Prime Mimi Prime Toshko Removed Mimi Last is Toshko
Gosho Pesho Misho Stefan Krasi 10	Removed Krasi Prime Pesho Prime Misho Removed Stefan Prime Gosho Removed Gosho Prime Misho Removed Pesho Last is Misho