# Homework: Stack & Queue / Set & Dictionary / File & Directory / Exception

Please submit as homework a single zip / rar / 7z archive holding the solutions (source code only) of all below described problems.

1. **Calculate Sequence with Queue**

We are given the following sequence of numbers:

* S1 = N
* S2 = S1 + 1
* S3 = 2\*S1 + 1
* S4 = S1 + 2
* S5 = S2 + 1
* S6 = 2\*S2 + 1
* S7 = S2 + 2
* …

Using the **ArrayDeque<E>** class, write a program to print its first 50 members for given N.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | 2 3 5 4 4 7 5 6 11 7 5 9 6 … |
| -1 | -1 0 -1 1 1 1 2 … |
| 1000 | 1000 1001 2001 1002 1002 2003 1003 … |

1. **Balanced Parentheses**

Given a sequence consisting of parentheses, determine whether the expression is balanced. A sequence of parentheses is balanced if every open parenthesis can be paired uniquely with a closed parenthesis that occurs after the former. Also, the interval between them must be balanced. You will be given three types of parentheses: (, {, and [.

{[()]} - This is a balanced parenthesis.

{[(])} - This is not a balanced parenthesis.

**Input Format:** Each input consists of a single line, S, the sequence of parentheses.

**Constraints:**   
1 ≤ lens ≤ 1000, where lens is the length of the sequence.   
Each character of the sequence will be one of {, }, (, ), [, ].

**Output Format:** For each test case, print on a new line "*YES*" if the parentheses are balanced. Otherwise, print "*NO*". Do not print the quotes.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| {[()]} | YES |
| {[(])} | NO |
| {{[[(())]]}} | YES |

## Periodic Table

You are given an n number of chemical compounds. You need to keep track of all chemical elements used in the compounds and at the end print all unique ones in ascending order:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Ce O  Mo O Ce  Ee  Mo | Ce Ee Mo O |
| 3  Ge Ch O Ne  Nb Mo Tc  O Ne | Ch Ge Mo Nb Ne O Tc |

## Count Symbols

Write a program that reads some text from the console and counts the occurrences of each character in it. Print the results in **alphabetical** (lexicographical) order.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| JLU CCST | : 1 time/s  C: 2 time/s  J: 1 time/s  L: 1 time/s  S: 1 time/s  T: 1 time/s  U: 1 time/s |  | Did you know Math.Round rounds to the nearest even integer? | : 9 time/s  .: 1 time/s  ?: 1 time/s  D: 1 time/s  M: 1 time/s  R: 1 time/s  a: 2 time/s  d: 3 time/s  e: 7 time/s  g: 1 time/s  h: 2 time/s  i: 2 time/s  k: 1 time/s  n: 6 time/s  o: 5 time/s  r: 3 time/s  s: 2 time/s  t: 5 time/s  u: 3 time/s  v: 1 time/s  w: 1 time/s  y: 1 time/s |

## Word Count

Write a program that reads a list of words from the file **words.txt** and finds how many times each of the words is contained in another file **text.txt**. Matching should be **case-insensitive**.

Write the results in file **results.txt**. Sort the words by frequency in descending order.

### Examples

|  |  |  |
| --- | --- | --- |
| **words.txt** | **Input.txt** | **Output.txt** |
| quick is fault | -I was quick to judge him, but it wasn't his fault.  -Is this some kind of joke?! Is it?  -Quick, hide here…It is safer. | is - 3  quick - 2  fault - 1 |

First read the **words1.txt** in order to retrieve the words you will be looking for, then read **text1.txt** to retrieve the text in which you will be searching for the words. Finally test your output against **01\_WordCount.txt**. Then read **words2.txt** and so on