

Western Kentucky University

High School Programming Contest

Please do not start until you are told to do so.

Problem 1: Sum of Divisors

You are to write a program which will input a positive integer n and output the number of divisors of n and the sum of those divisors. (Note: by divisor we mean a positive integer divisor – this includes 1 and n .)

Sample Run:

Please give me a positive integer: 10

10 has 4 divisors and their sum is 18.

Test Data:

12

15

32

109

Problem 2: Sum of Base-K Digits

You are to input two positive integer values, n and k , where k is greater than or equal to 2. The program should output the sum of the digits in the base k representation of n .

Sample Run:

Please give me the value of n : 27

Please give me the value of k : 4

The sum of the digits in the base 4 expansion of 27 is 6.

Test Data:

42 3

56 5

100 10

Problem 3: Program Profiler

You are to write a program which will accept from the user the number of lines in a program and the program itself (all entered interactively). It should then compute and print out the number of blank lines (containing only whitespace – spaces and tabs), the number of comment lines (the first two nonblank characters in the line are //), and the number of code lines (all the remaining lines).

Sample Run:

```
How many lines? 7
// Third sample problem solution
// Here it comes
```

```
public class Frog
{
}
```

There are 2 blank lines, 2 comment lines, and 3 code lines.

Test Data:

```
// sample solution for problem 1
// calculate the sum divisors
```

```
import java.io.*;
```

```
public static void main(String[] args) throws IOException
{
    // read two numbers in
}
```

Problem 4: Run-Length Encoder

You are to write a run-length encoder for the following byte-oriented encoding format. Runs of the same, repeated byte in the original file are replaced with a count of the number of repeated bytes, stored as a byte, followed by the repeated byte itself. For example, the string “HELLO”, would be encoded as follows:

| | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|
| original | H | E | L | L | O | | | |
| encoded | 1 | H | 1 | E | 2 | L | 1 | O |

Each letter is actually the ascii value of the character.

Note that there is no reason to use a count of zero (0), so we can assume a count of 0 actually represents 256. Also note that if there are more than 256 letters in a row, multiple counts are needed since byte only store values up to 255. In such cases, all but the final count values should be 0. See example below, which involves encoding a string of 1000 H's:

| | | | | | | | | |
|----------|---|---|---|---|---|---|-----|-------|
| Original | H | H | H | H | H | H | H | |
| | 0 | H | 0 | H | 0 | H | 232 | H |

A utility named “viewer” is provided in your disk to view the content of a file. To view the content, simply type in “viewer filename” from DOS command prompt.

Sample Run 1:

```
HELLO (content in input file program4test1.txt)
(content in output file program4out.txt)
$viewer program4out1.txt
viewer program4out1.txt
001 072 001 069 002 076 001 079
```

Sample Run 2:

```
H.....(1000 times program4test2.txt)
(content in output file program4out.txt)
$viewer program4out2.txt
viewer program4out2.txt
000 072 000 072 000 072 232 072
```

[ASCII table is provided for your reference]

Problem 5: Sales of Cookies

There is a local school attempting to raise funds by selling boxes of cookies. Your program should take a data file containing information about how many boxes of cookies have been sold by individual students and which class that student belongs to.

Your program should:

1. print the total number of boxes sold and display the amount sold for each class in descending order.
2. print the average number of boxes sold per student in each class in descending order.

There are only 10 classes in the school, but there is no limit on the number of students in each class. The input file is named “program5test.txt” and is provided in the disk in your package. The file is in a very simple format: Each student’s sales are described by two lines in the file. The first line specifies the class number for the student (a number between 1 and 10 inclusive). The second line indicates how many boxes of cookies that the student sold (an integer). You may assume that the file contains an even number of lines, but there is no limit on how long the file can be,

In the example below, we are assuming only three classes. You must assume there are 10 classes in your program.

Sample Run:

Input file:

| File content | Comment (not part of the file) |
|--------------|--------------------------------|
| 2 | a student from class 2 |
| 12 | sold 12 boxes |
| 1 | a student from class 1 |
| 0 | sold 0 boxes |
| 3 | a student from class 3 |
| 15 | sold 15 boxes |
| 2 | a student from class 2 |
| 6 | sold 6 boxes |

Output:

Total boxes sold: 33

Classes ordered by sales:

| Class number | boxes sold |
|--------------|------------|
| 2 | 18 |
| 3 | 15 |
| 1 | 0 |

Classes ordered by average:

| Class number | average boxes per student |
|--------------|---------------------------|
| 3 | 15 |
| 2 | 9 |
| 1 | 0 |