CS 121 – Some practice problems

Work in small groups, not by yourself. Write down in English pseudocode how to solve each problem.

1. The Butcher – Suppose a local butcher prepares packages of ground beef for sale, and every package has a weight between 1.00 and 2.00 pounds. Today, there is a sale: $1 off any purchase of ground beef that totals 4 or more pounds. In order to take advantage of this sale, we'll need to buy 3 packages of beef, but making sure their total weight is closest to 4 pounds without going under. In the meat case there are many packages of ground beef, and your job is to figure out which 3 packages have the smallest sum that's greater than 4.

   You may assume that the input is coming from a file called beef.txt. On each line of this file there is a real number representing the weight of a package of ground beef. You may assume that there are at most 100 numbers in the file.

   Your output should go to the screen, and it must tell the user which packages to buy, and what their total weight is. For purposes of identification, you may assume the packages are numbered from 1 up to the number of packages.

   Note – You must format your output weight to the nearest hundredth of a pound.

   Example beef.txt input file:
   ```
   1.71
   1.16
   1.43
   1.55
   1.08
   ```

   Example output:
   ```
   Buy packages 3, 4 and 5 for a total weight of 4.06 pounds.
   ```

2. Printing Primes – Write a program that prints some prime numbers to the screen. First, ask the user to enter a positive integer. (Error checking of input is not required.) Then, the program should print all the prime numbers that are less than or equal to this integer. These prime numbers should be printed 5 per line (except the last line if you run out of prime numbers to print). Allocate a horizontal field width of 8 characters to print each prime number.

   Note – a "prime number" is an integer that is divisible by exactly two numbers: 1 and itself.

   Example Input:

   Please enter a positive integer: **53**

   Example Output:
```
       2       3       5       7      11
      13      17      19      23      29
      31      37      41      43      47
      53
```

3. Duplicate word – Write a program that reads a text file (containing at most 1000 words), and prints out all the words that appear two or more times in the file. You may assume that punctuation marks such as the period, comma, question-mark and exclamation point are not part of words.

   Your program should not be case sensitive. In other words, it shouldn't matter if letters are capital or lowercase: "cat" and "Cat" should be considered the same word.

   You may assume that the input file is words.txt. The output goes to the screen: just list the words that are duplicated. This list will be presented *in alphabetical order*. Be sure that you print each duplicated word just *once*, even it appears many times in the file.

   Example input file:

   How much wood could a woodchuck chuck,
   if a woodchuck could chuck wood?

   Example output:
```
a
chuck
could
wood
woodchuck
```

4. Where can the Knight go?
Of all the chess pieces, the knight has the most interesting way of moving. Rather than just moving in a straight line or diagonal… the knight can only move in an L-shaped manner like this: to another square that is exactly 2 squares away in one direction **and** 1 square in a perpendicular direction. For example, a typical situation looks like this:

		*		*			
	*				*		
			K				
	*				*		
		*		*			

In this picture, the knight (K) is at row 4, column 4, and the asterisks (*) represent all the possible places where the knight can move to. The upper left corner of the chessboard is position 1,1. When we write the coordinates of some position on the board, the first number is the row number, and the second number is the column number. Rows go across (horizontal), and columns go down (vertical).

The input will come from a file called knight.txt, and the output should go to the screen.

The first line of input will be the number of starting positions that you need to handle. Each subsequent line of input will contain the row & column coordinates of a starting position separated by a comma (for example: 3,5). For each of these starting positions, your program needs to determine all possible destinations of the knight. These possible destinations should all be printed on the same line, separated from each other by 2 spaces. Introduce each set of possible destinations with the phrase "From … you can go to: " where … is the starting position.

Additional notes:

- The rows and columns are numbered 1-8.
- When printing the possible destinations from a particular starting position, the order does not matter.
- You may assume that the knight is the only chess piece on the board

Example Input:
```
2
3,5
4,8
```

Example Output:
```
From 3,5 you can go to:  1,4  1,6  2,3  2,7  4,3  4,7  5,4  5,6
From 4,8 you can go to:  2,7  3,6  5,6  6,7
```

5. “Seeing Stars”
   Design a program that will read a file called stars.txt. This file contains a list of numbers (you may assume they are positive integers). The numbers will appear one per line. Assume you don’t know how many numbers are in the file; in fact the file may be empty. For each number you encounter, print to the screen that number of asterisks, right justified. For example, if the file contains the numbers 3, 6 and 2, you should print:

```
   ***
******
    **
```

   You may assume that the largest number will be 20 or less.

6. “The Price Is Right”
   In this program, we will simulate the pricing game that appears at the beginning of every show of “The Price Is Right”. Four contestants have to guess the price of some item. Ask the user for the name of the input file, and then read this file. It contains 5 lines of information. The first line will be an item up for bids – its name and retail price. The other four lines of input will each contain a contestant name and their guess. You may assume that a comma will separate name from price.

   In your output, you are to announce the winner of the game – the person who guessed the price closest without going over. You also need to handle special cases. If the 4 contestants all overbid, there is no winner, and you need to announce, “You all overbid!” You may assume that all 4 bids are different. If anyone guesses the price exactly right, announce this as well.

   Here is some example input so you can see its format:

```
Four-poster bed, 1293
Julie, 800
John, 1000
Anne Marie, 1550
Sam, 1
```

7. “Is It Proper”
   Design a program that will read a text file called proper.txt. The program should be written so that it doesn’t matter how many lines there are or how long each line is. Your job is to count all the words that begin with a capital letter. Print each of these capitalized words out as you encounter them, and at the end you should print the number of capitalized words.

8. “Double Take”
   Design a program that will read a text file called double.txt. The program should be written so that it doesn’t matter how many lines there are or how long each line is. Your job is to all instances of repeated lines. In other words, tell the user which pairs of lines are the same. However, you are to *ignore* blank lines. (Of course, all blanks lines are the same, but this would not be useful information for the user.) For example, if the input file looks like this:

```
Dog
Cat
Tree
Cat
Dog
Dog
```

   then your program should say:

```
lines 1 and 5 are the same
lines 1 and 6 are the same
lines 2 and 4 are the same
lines 5 and 6 are the same
```

   (Don’t worry about redundant messages.)

9. “Tennis Anyone”
   Suppose you are looking for some land to build a house. Each neighborhood in town has somewhat different size lots for its houses. You want to be sure the lot you buy is big enough to accommodate a tennis court. Assume that a tennis court needs a rectangular area measuring at least 55 by 112 feet. Plus, the tennis court may not occupy more than half the entire lot, because we also need room for the house. Design a program that will read a text file called tennis.txt. This file will contain a list of locations and lot dimensions.

   Each line of the input file looks like this format:

   “Constantia Park, 100 by 120”. There will be a comma separating the name of the neighborhood and the dimensions, and the word “by” will separate the two dimensions.

   In your output, print the list of neighborhoods that have lot sizes big enough for a tennis court.

10. “Number Find”
    The input file number.txt contains 10 lines of data, and on each line there are 10 digits. Your job is to find the largest 5-digit number that appears on any row (as read from left to right), and also the largest 5-digit number that appears in any column (as read from top to botton). Print these two values to the screen. Also tell the user where these numbers were found, i.e. which row and which column.