

**Java program:** Prob01.java

**Input File:** Prob01.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Little Johnny would like help determining the value of the money in his piggy bank. He will provide you with the number of quarters, dimes, pennies, nickels, and half dollars he has in the bank. You need to tell him the total value. And no cheating little Johnny! The input file will contain a list of coins found in the piggy bank and the number of those types of coins. Coins will be in no particular order. The task is to determine the amount of money represented by the group of coins in the bank.

**Program Input**

The file Prob01.in.txt will contain a list of items and their quantities. There will only be one item and quantity pair per line, although duplicate items are allowed on different lines. Items and quantities will be separated by an equal sign with no spaces. There may also be non-coin items in the list that you should ignore. Valid coin names are: HALFDOLLAR, QUARTER, DIME, NICKEL, and PENNY.

**Example Input:**

QUARTER=31

DIME=5

NICKEL=2

DIME=7

HALFDOLLAR=0

PENNY=157

**Program Output**

Your program’s output should display the total value of the money in the bank in standard US currency format, using a dollar sign before the dollar amount and two decimal places for the number of cents.

**Example Output:**

$10.62

**Java program:** Prob02.java

**Input File:** Prob02.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

A neighborhood street has an equal number of houses on both sides of the street. Every house receives exactly one newspaper (either the Times or the Herald). You will be told how many houses on one side of the street subscribe to the Times, and how many houses on the other side of the street subscribe to the Herald. Your task is to figure out which newspaper has the bigger subscribership and how many more subscribers that paper has.

**Program Input**

The file Prob02.in.txt will contain subscriber data. Each line in the file describes a new block, and will contain two numbers separated by a space. The first number is the number of Times subscribers on one side of the street, and the second number is the number of Herald subscribers on the opposite side of the street.

**Example Input:**

22 9

6 14

12 12

**Program Output**

If your program calculates that the number of subscribers is equal for both newspapers, it should output the following message:

Times and Herald have the same number of subscribers

If your program calculates that the number of subscribers is not equal for both newspapers, it should output the following message:

[Newspaper name] has [total difference] more subscribers

Where

[Newspaper name] = "Times" or "Herald" (excluding quotes)

**Example Output:**

Times has 26 more subscribers

Herald has 16 more subscribers

Times and Herald have the same number of subscribers

**Java program:** Prob03.java

**Input File:** Prob03.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Write a program that will take a list of integers and determine whether they are in ascending, descending or random order. Integers can be positive or negative. If you encounter a non-integer value, then the entire line should be considered invalid input.

**Program Input**

The file Prob03.in.txt will contain some number of lines of characters separated by spaces. Each line should be treated as a new input set.

**Example Input:**

5 6 9 13 17 15

1 2 3 4 5

-1 -2 -3 -4 -5

1 2 3 4 =5

**Program Output**

Your program should output one of the following messages:

* The numbers are in random order
* The numbers are in ascending order
* The numbers are in descending order
* The input was invalid

**Example Output:**

The numbers are in random order

The numbers are in ascending order

The numbers are in descending order

The input was invalid

**Java program:** Prob04.java

**Input File:** Prob04.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

You have been asked to decode a secret message. A General Decoder takes a set of input strings. The first string is the English alphabet encoded in the order needed to decode the subsequent strings of numbers.

For example, the sequence:

18-5-1-4 20-8-9-19 13-5-19-19-1-7-5

Can be decoded by applying the alphabet order:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

To read:

READ THIS MESSAGE

Your task is to write a program that will take as inputs a decoding key and an encoded message, and output the decoded message.

**Program Input**

The file Prob04.in.txt will contain the alphabet decoding key on the first line followed by any number of lines of encoded messages. Numbers correspond to the position of the letter they represent in the decoding key, and are separated by either a dash or a space. Dashes are used to separate letters within a word, and spaces denote word boundaries. There will be no numbers less than 1 or greater than 26.

**Example Input:**

AEIOUYBCDFGHJKLMNPQRSTVWXZ

22-24-4 20-4-1-9-21 9-3-23-2-20-11-2-9 3-17 1 24-4-4-9 1-17-9 3

3 22-4-4-14 22-12-2 4-17-2 15-2-21-21 22-20-1-23-2-15-2-9 7-6

1-17-9 22-12-1-22 12-1-21 16-1-9-2 1-15-15 22-12-2 9-3-10-10-2-20-2-17-8-2

**Program Output**

Your program should output the decoded message.

**Example Output:**

TWO ROADS DIVERGED IN A WOOD AND I

I TOOK THE ONE LESS TRAVELED BY

AND THAT HAS MADE ALL THE DIFFERENCE

**Java program:** Prob05.java

**Input File:** Prob05.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

With interest rates at historic lows, many people are seeking to refinance their mortgages. But, in order to do that, they have to know how much they owe on their current mortgage. Your task is to write a program that will figure out the remaining principal due on a fixed rate loan. Here are some definitions and formulas pertaining to loans that you might find useful when writing your program:

A fixed-rate loan has equal monthly payments. To calculate the monthly payment M, you can use this formula:

Where:

* P is the original principal of the loan
* J is the monthly interest rate of the loan (J = I / 1200, if I is the yearly interest rate)
* N is the number of months it takes to pay off the loan

Payments are made at the end of each month. To calculate the remaining principal for a given month x, you can use this formula:

Where:

* Px-1 is the remaining principal for the previous month.
* M is the monthly payment amount
* J is the monthly interest rate

**Program Input**

The file Prob05.in.txt will contain loan information, with each new line denoting a new loan. Values on each line will be separated by spaces. The file will be in the following format:

OriginalPrincipal YearlyInterestRate TotalMonths MonthsPaid

The Original principal and yearly interest rate values will be floating point numbers, and the total months and months paid values will be integers.

**Example Input:**

100000 4.75 60 37

150000 6.00 120 18

**Program Output**

Your program must print the remaining principal amount (rounded to the nearest cent) after making the Xth monthly payment.

**Example Output:**

41157.60

132805.23