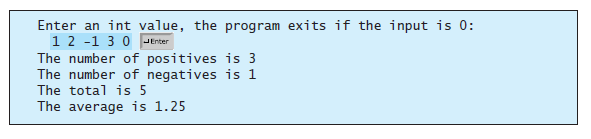
1. (*Counting positive and negative numbers and computing the average of numbers*)

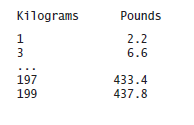
Write a program that reads an unspecified number of integers, determines how

many positive and negative values have been read, and computes the total and

average of the input values (not counting zeros). Your program ends with the input 0. Display the average as a floating-point number. Here is a sample run:

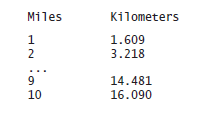


1. Suppose that the tuition for a university is **$10,000** this year and tuition increases **7%** every year. In how many years will the tuition be doubled?
2. (*Conversion from kilograms to pounds*) Write a program that displays the following table (note that **1** kilogram is **2.2** pounds):



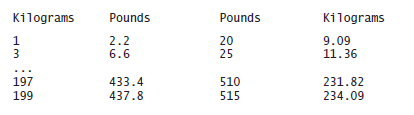
1. (*Conversion from miles to kilometers*) Write a program that displays the following

table (note that **1** mile is **1.609** kilometers):



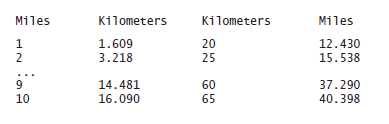
1. *Conversion from kilograms to pounds*) Write a program that displays the following

two tables side by side (note that **1** kilogram is **2.2** pounds):



1. (*Conversion from miles to kilometers*) Write a program that displays the following

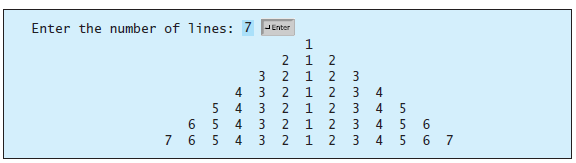
two tables side by side (note that **1** mile is **1.609** kilometers):



1. (*Financial application: computing future tuition*) Suppose that the tuition for a university is **$10,000** this year and increases **5%** every year. Write a program that computes the tuition in ten years and the total cost of four years’ worth of tuition starting ten years from now.
2. (*Finding the two highest scores*) Write a program that prompts the user to enter the number of students and each student’s name and score, and finally displays the student with the highest score and the student with the secondhighest score.
3. (*Finding numbers divisible by* **5** and **6**) Write a program that displays all the numbers from **100** to **1000**, ten per line, that are divisible by **5** and **6**.
4. (*Finding numbers divisible by* **5** or **6**, *but not both*) Write a program that displays all the numbers from **100** to **200**, ten per line, that are divisible by **5** or **6**, but not both.
5. (*Finding the smallest* **n** such that  **n2 >** 12,000) Use a **while** loop to find the smallest integer **n** such that n2is greater than 12,000.
6. (*Finding the largest* **n** such that  **n3 >** 12,000) Use a **while** loop to find the smallest integer **n** such that n3is less than 12,000.
7. (*Displaying the ACSII character table*) Write a program that prints the characters in the ASCII character table from **‘!'** to **‘~'**. Print ten characters per line.
8. (*Computing the greatest common divisor*) Write a program to find the greatest common divisor of two integers **n1** and **n2** is as follows: First find **d** to be the minimum of **n1** and **n2**, then check

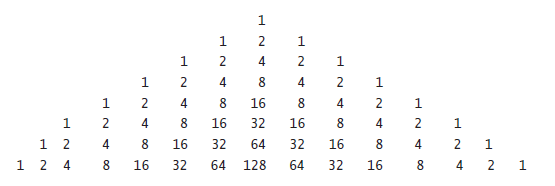
whether **d**, **d-1**, **d-2**, …,**2**, or **1** is a divisor for both **n1** and **n2** in this order. The first such common divisor is the greatest common divisor for **n1** and **n2**. Write a program that prompts the user to enter two positive integers and displays the gcd.

1. (*Finding the factors of an integer*) Write a program that reads an integer and displays all its smallest factors in increasing order. For example, if the input integer is **120**, the output should be as follows: **2**, **2**, **2**, **3**, **5**.
2. (*Displaying pyramid*) Write a program that prompts the user to enter an integer from **1** to **15** and displays a pyramid, as shown in the following sample run:



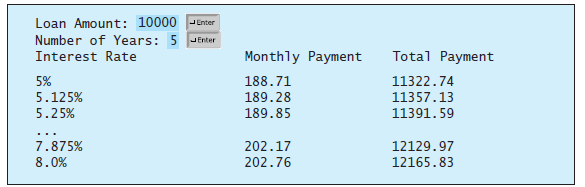
1. (*Printing numbers in a pyramid pattern*) Write a nested **for** loop that prints the

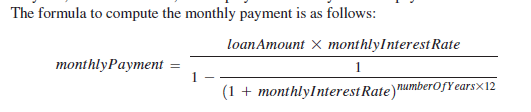
following output:



1. (*Financial application: comparing loans with various interest rates*) Write a program

that lets the user enter the loan amount and loan period in number of years and displays the monthly and total payments for each interest rate starting from 5% to 8%, with an increment of 1/8. Here is a sample run:





1. (*Obtaining more accurate results*) In computing the following series, you will obtain

more accurate results by computing from right to left rather than from left to right:



Write a program that compares the results of the summation of the preceding series, computing from left to right and from right to left with **n 50000**.

1. (*Summing a series*) Write a program to sum the following series:

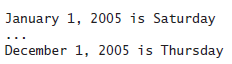


1. (*Computing π* ) You can approximate π by using the following series:



Write a program that displays the value for **i 10000**, **20000**, and **100000**.

1. (*Displaying leap years*) Write a program that displays all the leap years, ten per line, in the twenty-first century (from 2001 to 2100).
2. (*Displaying the first days of each month*) Write a program that prompts the user to enter the year and first day of the year, and displays the first day of each month in the year on the console. For example, if the user entered the year **2005**, and **6** for Saturday, January 1, 2005, your program should display the following output. (note that Sunday is **0**):



1. (*Financial application: compound value*) Suppose you save **$100** *each* month into a savings account with the annual interest rate **5%**. So, the monthly interest rate is **0.05 / 12 = 0.00417**.

After the first month, the value in the account becomes

100 \* (1 + 0.00417) = 100.417

After the second month, the value in the account becomes

(100 + 100.417) \* (1 + 0.00417) = 201.252

After the third month, the value in the account becomes

(100 + 201.252) \* (1 + 0.00417) = 302.507

and so on.

Write a program that prompts the user to enter an amount (e.g., **100**), the annual

interest rate (e.g., **5**), and the number of months (e.g., **6**) and displays the amount in the savings account after the given month.

1. (*Financial application: computing CD value*) Suppose you put $10,000 into a CD with an annual percentage yield of 5.75%.

After one month, the CD is worth

10000 + 10000 \* 5.75 / 1200 = 10047.91

After two months, the CD is worth

10047.91 + 10047.91 \* 5.75 / 1200 = 10096.06

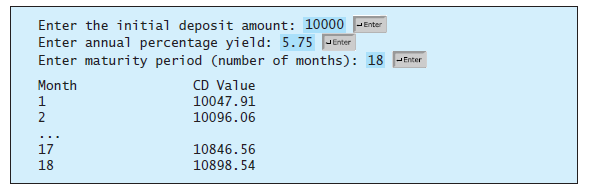
After three months, the CD is worth

10096.06 + 10096.06 \* 5.75 / 1200 = 10144.43

and so on.

Write a program that prompts the user to enter an amount (e.g., **10000**), the annual

percentage yield (e.g., **5.75**), and the number of months (e.g., **18**) and displays a table as shown in the sample run.



1. (*Perfect number*) A positive integer is called a *perfect number* if it is equal to the

sum of all of its positive divisors, excluding itself. For example, **6** is the first perfect

number because **6 = 3 + 2 + 1**. The next is **28 = 14 + 7 + 4 + 2 + 1**. There are four perfect numbers less than **10000**. Write a program to find all these four numbers.

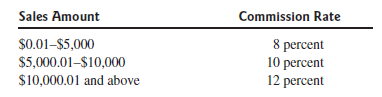
1. (*Decimal to binary*) Write a program that prompts the user to enter a decimal

integer and displays its corresponding binary value.

1. (*Decimal to hex*) Write a program that prompts the user to enter a decimal integer

and displays its corresponding hexadecimal value.

1. (*Financial application: finding the sales amount*) You have just started a sales job in a department store. Your pay consists of a base salary and a commission. The base salary is $5,000. The scheme shown below is used to determine the commission rate.



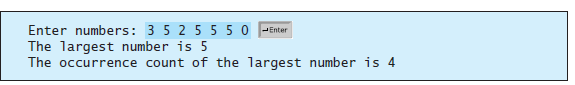
Your goal is to earn $30,000 a year. Write a program that finds out the minimum

amount of sales you have to generate in order to make $30,000.

1. (*Occurrence of max numbers*) Write a program that reads integers, finds the largest of them, and counts its occurrences. Assume that the input ends with number **0**. Suppose that you entered **3 5 2 5 5 5 0**; the program finds that the largest is **5** and the occurrence count for **5** is **4**.

(*Hint*: Maintain two variables, **max** and **count**. **max** stores the current max number, and **count** stores its occurrences. Initially, assign the first number to **max** and **1** to **count**. Compare each subsequent number with **max**. If the number is greater than **max**, assign it to **max** and reset **count** to **1**. If the number is equal to

**max**, increment **count** by **1**.)



31) (*Financial application: finding the sales amount*) Rewrite Exercise 29 as follows:

Use a **for** loop instead of a **do-while** loop.

Let the user enter **COMMISSION\_SOUGHT** instead of fixing it as a constant.