Problem #1
BASE NEGATIVE 2
Novice

Source file: problem1.extension

1, 110, 111, 100, 101, 11010, 11011, 11000, 11001, 11110

This is how you count to ten in base –2. The digits of base –2 are 0 and 1 and each digit position stands for a successive power of –2: 1, -2, 4, -8, …. One can express all positive and negative numbers in this way without having to append a negative sign onto the whole quantity. –5 in base –2 is represented by 1111.

The purpose of this program to determine the base ten equivalent of a base –2 number input to the program.

INPUT:  Your program should accept a base -2 number at a suitable prompt. Your program should continue to accept input until a zero is entered. At this point your program should terminate. (You may assume that no number will require more than 32 bits.)

OUTPUT:  Your program should output the result in the following form:

nnnnnn = dddddd

EXAMPLE:  Enter an number: 1101100
1101100 = 28
Enter an number: 100010
100010 = -34
Enter an number: 0
Here is another sequence problem. Each number of the sequence is formed by the sum of the previous number in the sequence plus the product of that number’s digits. The initial number in this sequence could be any integer.

An example sequence would be: 13, 16, 22, 26, 38, 62 starting with 13. Note that 26 is the 4\textsuperscript{th} item in the sequence.

The purpose of this problem is to determine the $k$\textsuperscript{th} item in this sequence starting with a given number, $n$.

**INPUT:** Your program should accept two integers separated by a space at a suitable prompt. The first integer will be $n$, the initial number of the sequence. The second integer will be $k$ which indicates the item of the sequence to output. Your program should continue to accept inputs until at least one of the entries is 0. At this point your program should terminate.

**OUTPUT:** Your output should be of the form:

\[\text{xxx is item #}k\text{ in the sequence.}\]

**EXAMPLE:**

Enter $n$ and $k$: **32 8**
102 is item #8 in the sequence.

Enter $n$ and $k$: **822 15**
1014 is item #15 in the sequence.

Enter $n$ and $k$: **999 9**
1840 is item #9 in the sequence.

Enter $n$ and $k$: **0 0**
The following is the first 7 rows of a triangle known as Pascal’s Triangle:

   1
  1 1
 1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1

Note that Pascal’s Triangle is formed by one’s as the first and last elements of each row and all the other numbers are the sum of the two elements from the row directly above the number to the left and right. As examples, the 2 in the 3rd row is the sum of the 1 and the 1 in the row directly above the 2 and the 20 in the 7th row is the sum of the 10 and the 10 from the row directly above the 20. Or in other words, the $k$th entry of row $n$ is formed by adding the $(k-1)^{st}$ and $k^{th}$ entries of row $n-1$. Also note that 2 is the 2nd entry of the 3rd row and 20 is the 4th entry of the 7th row. The 1 at the apex of the triangle is the 1st entry of the 1st row.

The purpose of this program is to output the $k^{th}$ entry of the $n^{th}$ row of Pascal’s Triangle, when given the values for $n$ and $k$. The maximum size for Pascal’s Triangle is 20 rows ($n \leq 20$).

INPUT: Your program should accept two integers, separated by a space, representing a row number ($n$) and an entry number ($k$) at an appropriate prompt. Your program should continue to accept two integers until two zeroes (0 0) are entered. At this point the program should terminate.

OUTPUT: Output the $n^{th}$ row, $k^{th}$ entry of Pascal’s Triangle on a new line.

EXAMPLE: Enter n,k: 5 4
4
Enter n,k: 3 1
1
Enter n,k: 8 4
35
Enter n,k: 0 0
In the attic of the house where I live, my brother found three abandoned logic machines, and converted them into Transmogrifiers: a Dogmatizer, a Categorizer, and a Rationalizer. The Dogmatizer would change one dog into one cat and one rat. The Categorizer would change one cat into two dogs and one rat. The Rationalizer would change one rat into three dogs and one cat. Each machine would also work in reverse, for example, the Dogmatizer would change one cat and one rat into one dog.

Starting with my cat, Domino, you will be given a sequence of operations (as single characters) representing the Transmogrifier to apply. Once the sequence of operations is terminated, you will output the number of cats, dogs, and rats that are now present.

The operation characters are:

- **d**: Use Dogmatizer to change one dog into one cat and one rat.
- **c**: Use Categorizer to change one cat into two dogs and one rat.
- **r**: Use Rationalizer to change one rat into three dogs and one cat.
- **D**: Use Dogmatizer to change one cat and one rat into one dog.
- **C**: Use Categorizer to change two dogs and one rat into one cat.
- **R**: Use Rationalizer to change three dogs and one cat into one rat.
- *****: Terminates operation sequence.

**INPUT:** Your program should accept a string of characters as described above at a suitable prompt. Your program should continue to accept input strings until a ‘q’ is entered. At this point the program should terminate.

**OUTPUT:** Your output should be the number of cats, dogs, and rats present after the input string has been processed according to the above rules. The form of the output should be:

```
Cats = XXX  Dogs = XXX  Rats = XXX
```

If an operation in the string can not be performed due to insufficient inputs for the operation, stop processing the string and output the error message: “Cannot continue, insufficient inputs, error processing operation X”, where X is the operation character causing the error.

**EXAMPLES:**

Enter operation string: cdcdcd*
Cats = 1  Dogs = 3  Rats = 6
Enter operation string: crdcRd*
Cats = 0  Dogs = 2  Rats = 1
Enter operation string: cdrCDR*
Cannot continue, insufficient inputs, error processing operation D
Enter operation string: q
A word chain is a daisy chain made from words. Two words, \( w_1 \) and \( w_2 \), may be linked if the last letter of \( w_1 \) is the same as the first letter of \( w_2 \). For example, ‘bar’ and ‘red’ may be linked. A word chain must consist of four words and be circular (that is, the last word should link to the first word). No word may be used more than once. For example, big-gap-pet-tab is a word chain.

You will be given four words, in random order, as input and you must determine whether those four words determine a word chain. If so, you will output the word chain. The words will be from 2 to 5 characters in length.

INPUT: Your program should accept four words, all entered on the same line, with a suitable prompt. Your program should continue to accept four words until the first word of entry is ‘quit’. At this point your program should terminate.

OUTPUT: If the four input words form a word chain, output the word chain with each word in the chain separated by hyphens. Answers may vary since you could choose to start your chain with any of the four words.

EXAMPLES: Enter words: pet big gap tab
            gap-pet-tab-big

            Enter words: nog hello onion graph
            hello-onion-nog-graph

            Enter words: hello graph onion waste
            No word chain possible

            Enter words: quit stop end finis
Problem #6
QUARTERBACK RATINGS
Novice

Source file: problem6.extension

The following is taken verbatim from the web site:
http://football.about.com/c/ht/03/03/How_Calculate_Quarterback_Rating1048560068.htm.
This algorithm is used to determine a quarterback’s rating so that quarterbacks may be compared statistically over a number of categories with other quarterbacks.

1. Divide a quarterback’s completed passes by pass attempts.
2. Subtract 0.3.
3. Divide by 0.2 and record the total. The sum cannot be greater than 2.375 or less than zero.
4. Divide passing yards by pass attempts.
5. Subtract 3.
6. Divide by 4 and record the total. The sum cannot be greater than 2.375 or less than zero.
7. Divide touchdown passes by pass attempts.
8. Divide by 0.05 and record the total. The sum cannot be greater than 2.375 or less than zero.
9. Divide interceptions by pass attempts.
10. Subtract that number from 0.095.
11. Divide that product by 0.04 and record the total. The sum cannot be greater than 2.375 or less than zero.
12. Add the four totals you recorded.
13. Multiply that total by 100.
15. The final number is your quarterback rating.

The purpose of this problem is to read in four sets of statistics for four different quarterbacks and order them by their rating from best to worst (or top to bottom).

INPUT: Your program should accept four sets of data, one for each quarterback, at an appropriate prompt. The data will be entered in the order: quarterback’s last name, passes attempted, passes completed, passing yards, touchdown passes, and interceptions. You may assume all the data is correct.

OUTPUT: Output in columns the quarterback’s last name and rating, one quarterback per line, in order of highest rating to lowest rating.

EXAMPLE: Enter data set 1: Tarkenton 6467 3686 47003 342 266
Enter data set 2: Kelly 1742 1032 12901 81 63
Enter data set 3: McMahon 1831 1050 13335 77 66
Enter data set 4: Anderson 4475 2654 32838 197 160

Kelly  82.7401
Anderson  81.858
Tarkenton  80.3544
McMahon  79.2156
Every book has an International Standard Book Number (ISBN). The 10 digit ISBN has a built-in "check" to ensure that the numbers have been correctly entered into the computer or that the numbers have been scanned properly by an optical scanner. If you multiply each digit of the ISBN from left to right with 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 in order and sum the products, you will obtain a result that is 0 mod 11. This scheme detects 100% of all single digit errors and 100% of all transposition errors. The “check” digit is the 10th digit of the ISBN. The drawback of this method is that in some cases the check digit is required to be 10, which is not a single digit. To maintain a uniform ten digit format for all books the character X is used to represent 10.

For example, the ISBN 0-669-19493-X satisfies the above formula as 0*10 + 6*9 + 6*8 + 9*7 + 1*6 + 9*5 + 4*4 + 9*3 + 3*2 + 10*1 = 275 = 0 mod 11 as one can easily show that 11 divides 275. Therefore, this number is coded correctly.

The purpose of this problem is to determine the digit to place in position 10 given the first 9 digits of the ISBN such that the resultant number is a valid ISBN number.

**INPUT:** Your program should accept the first 9 digits of an ISBN in the form: d-ddd-ddddd at a suitable prompt. Your program should continue to accept ISBNs until an ISBN of 9-999-99999 is entered. At this point your program should terminate.

**OUTPUT:** Output the complete ISBN in the form: ISBN = d-ddd-ddddd-d.

**EXAMPLE:**
Enter partial ISBN: **0-671-79499**
Enter partial ISBN: **0-812-91867**
ISBN = 0-812-91867-3
Enter partial ISBN: **9-999-99999**
Problem #8
FACTORIAL DIGIT FREQUENCY
Novice

Source File:  problem8.extension

The purpose of this program is to compute the digit frequencies of the factorials from 1 to k, where 1 ≤ k ≤ 12. Recall that a factorial of a number n, denoted by n!, is the product of the integers from 1 to n. For example, 9! = 362880. 9! contains 1 zero, 1 two, 1 three, 1 six, and 2 eights.

INPUT: Your program should accept a integer for k at a suitable prompt. Your program should continue to accept input until a 0 is entered. At this point your program should terminate.

OUTPUT: Output a table where the first column consists of the digits from 0 to 9 and the second column is the frequencies of those digits in the factorials from 1! to k!.

EXAMPLE: Enter k: 4

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

Enter k: 0