Returning Values From Functions

Functions Returning a Single Value

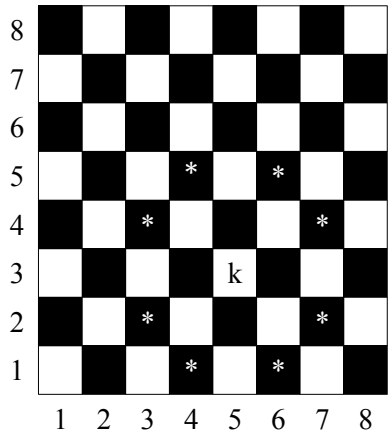
1. Given three values, determine the sum of the smallest and largest values, and return this value to your main program. Display this value.
2. Given a positive integer, determine the number of 7s it contains. Return this value to your main program. Generate 20 random integers between 1 and 1 000 000, and determine the total number of 7s from all numbers.
3. Given a string, determine the number of "words" in it. Return the count to your program.
4. A perfect number is one that is the sum of its proper divisors. For example, 28 has proper divisors 1, 2, 4, 7 and 14, and $1+2+4+7+14=28$, so 28 is a perfect number. Write a function that determines whether a number is perfect or not, and returns `True` or `False`. Use your program to display all perfect numbers between 1 and 1 000 000.

Functions Returning Multiple Values

5. Write a function that takes the radius, *r*, of a circle, and calculates its circumference and its area. Return these values to your main program, then print them.
6. Given a value of *n*, roll two dice with *n* sides and return these two values. Roll the dice 20 times and display the values of the dice from the main program.
7. Write a function, *slope*, that calculates the slope of the line through two points, (p, q) and (r, s). Represent the slope as a fraction (unreduced is fine), then return the numerator and denominator to your main program. Write a second function, *y_intercept*, that takes the numerator, denominator, and coordinates of a point (p, q), and calculates the *y*-intercept of the line. Return the *y*-intercept. Display the equation of the line, $y=mx+b$, from your main program.

Challenge

8. In the game of chess, a knight moves in an L-shaped pattern, moving either two squares horizontally and one square vertically, or two vertically and one horizontally. The diagram at right shows the eight possible moves (marked with *) that a knight at square (5, 3) can make. It is possible, with some effort, for a knight to travel to *any* square on an 8x8 chessboard. For instance, the knight at (5, 3) can travel to (4, 7) by moving to (4, 5), then to (2, 6), then to (4, 7). This trip takes a total of 3 moves.

   Write a program that generates *different* random starting and finishing locations for a knight. After displaying the starting and finishing locations, the user should enter the coordinates of a new square to which the knight should move. If the move is valid, the knight should move to that square. Checking for a valid move should be handled by a function, *get_valid_move*, which takes the current position and potential coordinates, and returns `True` or `False` if the move is valid or not. The user should enter coordinates until the knight reaches the finishing location, upon which the program should display the number of moves made.