Problem Set 3

Functions

1. Implement the following function:

    a) *factorial (n)*-- which takes a number as parameter and return its factorial.

    Call the above function. Initialize an integer variable from the user and pass it as parameter to the above function. Then display the factorial.

2. Modify the factorial program to call the factorial function 10 times while passing a user defined integer each time and displaying the corresponding factorial.

3. An integer number is said to be a perfect number if the sum of its factors, including 1 (but not the number itself), is equal to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write a function *perfect* that determines whether parameter *number* is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect.

4. An integer greater than 1 is said to be prime if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime numbers, but 4, 6, 8 and 9 are not.
    i) Write a function that determines whether a number is prime.
    ii) Use this function in a program that determines and prints all the prime numbers between 2 and 1,000.
    iii) Initially, you might think that n/2 is the upper limit for which you must test to see whether a number is prime, but you need go only as high as the square root of n. Rewrite the program and run it both ways to show that you get the same result.

5. Implement the following function:

    a) *factors (n)* –displays all factors of the number *'n'* passed as parameter.

    b) *display(start, end)* –this function takes the start and end of a range as parameter and calls the above function *factors()* to display the factors of all numbers in that range.

    Now call the function *display()* in a program. Take the start and end of a range from the user and pass it to the function to display the factors.