1. Let's prctice writing functions. Make a program where you define functions that

   (a) print a message to the screen.

   (b) reads an integer from the user and then returns it

   (c) takes as an argument a realnumber x and returns $1/(1-x)$ (Ex 4.4)

   (d) takes two real numbers as arguments and returns the one that is greater.

   Write a main function where you test your functions.

2. (H4T3) Make a program that asks for a real number x and an integer n, and calculates the sum

$$\sum_{k=0}^{n} \frac{x^k}{k!}$$

3. Write a function called, say, exps, that calculates the sum of Ex. 2. The function should take x and n as its arguments and give the sum as its return value. Compare the results of your function to those of exp(x). Print to screen the values of your function, exp(x) and their difference. To use exp(x) (it calculates e^x) you need to include the #include <math.h> directive.

4. Write a program that calculates the growth of an amount of money in a bank account. The growth should be calculated in its own function using a for-loop. The user inputs the initial capital, the interest rate and the savings time in years. The capital grows each year according to the formula:

$$end\ of\ year\ capital = beginning\ of\ year\ capital \cdot \frac{100 + interest\ rate}{100}$$

   The function could be of the form calcinterest(double initcapital, double interestrate, int savingstime); The function should return the amount of money on the account at the end of the given time.

5. Extra: Change the summation in Ex. 2 so that the sum is computed until adding new terms does not change the sum.

6. Extra (evil): Functions may call themselves. This kind of functions are called recursive and in some programming languages recursive functions are a common way to program a loop. Calculate the factorial using a recursive function. As an aid you can use the follwing definition for the factorial (denoting $n! = f(n)$):

$$f(n) = \begin{cases} 1, & n = 0 \\ n\,f(n-1), & n > 0 \end{cases}$$