In class: Indicate in table below when you are done with each exercise

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exercise | 1 | 2 | 3 | 4 | 5 | 6 |
| Done? |  |  |  |  |  |  |

## Exercise 1: Write a function called sqrtsum that takes as input two arbitrary real numbers, x1 and x2 and returns two parameters: the sum of x1 and x2 and the square root of the sum of x1 and x2 (use the built-in matlab function sqrt). Include help lines that specify what the function does, and what the input and output arguments are.

Write the function definition line:

Write the help lines:

Write the body of the function:

## Exercise 2: Write down the function call from the main script if you want to find the “sqrtsum” of 255 and 73.5

**Exercise 3:** Now implement a check within the function that causes the program to terminate if the sum of x1 and x2 is negative. You can use error(‘hello’) to cause the function to stop and print an error message with the word ‘hello’.

## Exercise 4: Now change your code in (3) to be a subfunction called checkinput that does the exact same thing as in (3) and is called by your main function sqrtsum. checkinput should take *all* the input parameters passed to sqrtsum.

Function call inside sqrtsum:

Function definition line and body (no need for help lines in this one this time!):

**Exercise 5:** Continue writing the body of the function addn that takes three parameters n, summax and maxiter and adds n to itself while the sum is less than or equal to summax or if the number of iterations is less than or equal to maxiter and then returns the sum.

function sumn = addn(n,summax,maxiter)

% sumn(n,summax,maxiter) adds the n to itself until one of the

% two conditions is reached:

% 1. Either sumn reaches summax, or

% 2. maxiter iterations is reached

%

counter = 0;

sumn = n;

while sumn <=(summax-n)

end

**Exercise 6**: What’s the problem?

Command window:

|  |
| --- |
| >> x = 5.1;  >> y=myfracpart(x)  Undefined function or variable 'x'.  Error in myfracpart (line 3)  y=x-fix(x)  >> |

File myfracpart.m:

|  |
| --- |
| function out=myfracpart(in)  % MYFRACPART returns fractional part of number  y=x-fix(x); |