Name:

1. [**Matching (30 points)**] Find the **best** match for each concept on the left-hand side with the phrase on the right. Note that there are more items in the right column than in the left, so some answers will not be used.

___ nested loop

___ `const int identifier`

___ `imshow`

___ `my_func(char[] id);`

___ `12/5`

___ debugging

___ `getline()`

___ `sum([2 2 2].^2)`

___ !!!2

___ script

A. a function that takes data from a stream until a newline

B. tell compiler to include a library header file for identifying

C. a sequence of expressions

D. an iterative routine that does not terminate

E. 2

F. 12

G. 36

H. an iteration within an iteration

I. a variable declaration with an immutable value

J. false

K. true

L. a process of systematically checking your beliefs about the syntax/semantics of a code

M. a function that takes a C-string as an input

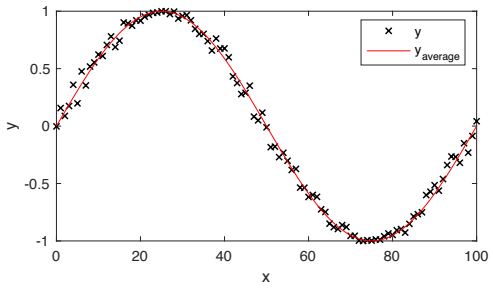
N. a function that displays truecolor RGB data

O. an Interactive MATLAB show

Name:

5. [**MATLAB: Short answers (20 points)**]

(a) Assuming you have a list of values `x` and `y` with average values `yav` at discrete intervals `xav`, write down **all** the expressions required to generate the following plot (NB: the points are black times (×) symbols and the line is red):




```
plot(x,y)
legends("x y", "y average")
ylabel("y")
xlabel("x")
xlimit(100)
ylimit(1)
```

(b) Explain what the following code does and write down what values of `indices` and `nletters` are printed at the end if `mychar = 'p';`?

```
name = 'Penelope Apple Penvelope Pitstop';
if(sum(name == mychar))
   indices = find(name == mychar);
   nletters = sum(name == mychar);
end
disp(indices); disp(nletters);
```

This one looks for mychar in the given name and it will print 7, 11, 12, 23, and 32 for indices, and 5 for n letters.

Name:

6. [**MATLAB: Vectorize this (30 points)**] For each of the functions below, write a vectorized version that avoids all use of explicit selection (`if, switch`) or iteration (`while, for`, recursive calls). You may leave out the function headers and just show the vectorized function bodies.

   (a)
```
function A = fun1(vec) % fun1: input is a column vector
jj=1;
for ii=1:length(vec)-1
   if vec(ii) < vec(ii+1)
      A(jj) = vec(ii) / abs(vec(ii));
      jj = jj + 1;
   end
end
end
```

   (b)
```
function result = fun2(mat) % fun2: input is a matrix
for ii = 1:size(mat,1)
   for jj = 1:size(mat,2)
      result(ii,jj) = mat(ii,size(mat,2) - jj + 1);
   end
end
end
```