A. V. Gerbessiotis — CS 101
Oct 21, 2014 — Fall 2014
PS 6 with Solutions — No points

---

**Problem 1.**
Write the output of the following two fprintf commands in the indented and boxed space underneath it that can accommodate 25 characters, one per box. (Three lines of output are provided; you might not need them all).

```
>> fprintf('matlab=%5d\n', 10);
```

```
----------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
----------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
----------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Third line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
----------------------------------------------------------------------------
```

```
>> fprintf('matlab=%-5d\n', 10);
```

```
----------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
----------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
----------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Third line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
----------------------------------------------------------------------------
```

**Problem 2.**
(a) What does the following MATLAB program print?

```
>>  x=50;
>>  y=70;
>>  z=90;
>>  if (x+ z  >  2*y)
       fprintf('this\n');
    elseif ( 2*x < z)
       fprintf('that\n');
    else
       fprintf('what\n');
    end
```

```
----------------------------------------------------------------------------
| | | | | | | | | | | | | | | | | | | | | | | | | |  % First line
| | | | | | | | | | | | | | | | | | | | | | | | | |
----------------------------------------------------------------------------
| | | | | | | | | | | | | | | | | | | | | | | | | |  % Second line
| | | | | | | | | | | | | | | | | | | | | | | | | |
----------------------------------------------------------------------------
```

(b) What does the following program print?

```
>> x=50;
>> y=70;
>> z= x+y;
>> if (z < 2*x)
      fprintf('%d\n',y);
   elseif (x == 2 * y)
      fprintf('%d\n',x+y);
   else
      fprintf('%d\n',x-y);
   end
--------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
--------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
--------------------------------------------------------------------------
```

(c) What does the following program print?

```
>> a=50; b=70;
>> if ( a < 10  & b > 20)
      fprintf('Hi\n');
   elseif ((a > 10) || (b< 20))
      fprintf('Ho\n');
   elseif ((a==10) && (b == 10))
      fprintf('He\n');
   else
      fprintf('Oh\n');
   end
--------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
--------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
--------------------------------------------------------------------------
```

**Problem 3.**
(a) Let `FourA` be a row vector. Write a correct piece of MATLAB code that counts the number of 1s in that vector. For example if vector `FourA` is [1 2 1 100 1.0] the answer will be 3. (Use an iterative structure, as opposed to the solution of PS4.)
(b) Let `FourB` be a row vector. Write a correct piece of MATLAB code that changes all 1s into 5s. For example if `FourB` is [1 2 1 100 1.0] the answer will be [5 2 5 100 5] afterwards. (Use an iterative structure, as opposed to the solution of PS4.)

**Problem 4.**
Let the factorial function $n!$, for any $n \geq 1$, be defined as the product $n \cdot (n-1) \cdot (n-2) \cdot 2 \cdot 1$. That is, $3! = 3 \cdot 2 \cdot 1 = 6$. Write a MATLAB `for` loop that computes $n!$ if a variable $n$ has already been defined in the program and holds a positive integer value whose factorial needs to be computed. The answer with the computed $n!$ should be stored in a variable named `factorial`. Note: You are not allowed to use the factorial or prod functions. You need to implement your own code using a for-loop or a while-loop (and not repeat a PS4 solution).

**Problem 5.**
A variable $n$ stores a positive integer value. Write MATLAB code that computes the sums

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n$$

and

$$\sum_{i=1}^{n} i^3 = 1^3 + 2^3 + 3^3 + \ldots + n^3$$

The result for the first sum should be made available in variable `sum1` and for the second in `sum2`.

**Problem 6.**
The following code is a non-elegant attempt to sort a vector of small integer numbers

```
% Code 1
a= [ 6 5 1 2 3 4];                % arbitrary vector of numbers
len =length(a);                   % find its length (i.e. number of elements)
disp(a);                          % print it
for ii=1:len                      % iterate over i
  [val index] = min(a);           % find the minimum of  a
  b(ii) = val;                    % the ii-th minimum value is stored in b(ii)
end
disp(b);
```

(a) Is there anything wrong with `Code 1`? Explain.
(b) If you think `Code 1` is wrong, how could you minimally modify it so that $b$ is indeed the sorted $a$? Explain. (You may assume that the input is filled with small integers, positive or negative, less than one billion.) **Hint:** To make things easier you are allowed to use the `min` or `max` function. This might cause some to argue about the `in-place` property of the resulting algorithm; ignore such nuisance.

```
% Code 3
a= [ 6 5 1 2 3 4];
len =length(a);
disp(a);
for ii=1:len-1
  [val index] = min(a(ii:len));
  if (index~= ii);  a(index)=a(ii); a(ii)=val; end
end
disp(a);
```

(c) Can you eliminate $b$ so that the output is in $a$? We say that such a sorting algorithm **sorts in-place** if it uses the same array/vector for input and output and whose remaining space requirements are constant (note that `len, ii, val ,index` are four, i.e. contsant, number of variables). Can `Code 3` be a reasonable solution to this problem? Explain or modify!
**Comment:** Ignore the effect of the use of the `min,max` function in the in-place property of the resulting algorihm.

Turn page for solutions

**Problem 1.**

```
 >> fprintf('matlab=%5d\n', 10);

-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
| m| a| t| l| a| b| =|  |  |  | 1| 0|  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
| >|> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Third line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------

 >> fprintf('matlab=%-5d\n', 10);

-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
| m| a| t| l| a| b| =| 1| 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
| >|> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Third line
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
```

**Problem 2.**

(a)

```
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|w |h |a |t |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|> |> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
```

(b)

```
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|- |2 |0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|> |> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
```

(c)

```
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % First line
|H |o |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  % Second line
|> |> |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
-------------------------------------------------------------------------
```

**Problem 3.**

(a)

```
len=length(FourA);
counter=0;
for ii=1:len
  if FourA(ii) == 1
      counter= counter+1;
  end
end
disp(counter);
```

The code above first computes the number of elements in the vector and stores the result in `len`. It initializes a counter named `counter` with zero. It then iterates overall elements of `FourA`; if an element is one, it increments the counter. The value of `counter` gives the number of ones.

(b)

```
len=length(FourB);
for ii=1:len
  if FourB(ii) == 1
      FourB(ii)=5;
  end
end
```

**Problem 4.**

The code is simple; we multiply the next value of an index variable `ii` to the product of the previous values computed in the previous iterations. After $ii-1$ iterations, `factorial` contains the product $1 \cdot 2 \cdot 3 \cdot (\texttt{ii}-1)$, and thus in the $ii$-th iteration, it will multiply $ii$ by this product thus generating $1 \cdot 2 \cdot 3 \cdot \ldots \cdot (\texttt{ii}-1) \cdot \texttt{i}$. This continues until $ii = n$ when the final result is computed and then displayed in the last line of the code below.

```
factorial = 1;
for ii=1:n
  factorial = factorial *ii ;
end
disp(factorial);
```

**Problem 5.**

(a)

```
sum1 =0;
for ii=1:n
  sum1 = sum1 + ii;
end
disp(sum1);
```

Of course a `sum(1:n)` would also work.

(b)

```
sum2 =0;
for ii=1:n
  sum2 = sum2 + ii^3;
end
disp(sum2);
```

One interesting property is that $\texttt{sum2} = \texttt{sum1} * \texttt{sum1}$, and $\texttt{sum1} = n(n+1)/2$. Of course a `sum2 = 1:n ; sum2 = sum2 . ^ 3` and then `sum2= sum(sum2)` would also work.

**Problem 6.**

(a) The fragment shown below will fill vector $b$ with the minimum value of $a$. For it to sort correctly it must fill the first position of $b$ with the minimum of all the numbers in $a$, the second position with the smallest of the remaining numbers (other than the already computed minimum), and so on. This is not that is being done in line 6. Thus the output vector $b$ for this given input will result in an all-one vector.

```
a= [ 6 5 1 2 3 4];
len =length(a);
disp(a);
for ii=1:len
  [val index] = min(a);
  b(ii) = val;
end
disp(b);
```

(b) The following code resolves these issues. As soon as we determine that $a(index)$, holds $val$, the minimum value, we store it in the output array and we replace it by a very large value (larger than the largest of the remaining keys). Such a value is `intmax` which is the largest integer (and is also larger than one billion, the maximum input value).

```
a= [ 6 5 1 2 3 4];
len =length(a);
disp(a);
for ii=1:len
  [val index] = min(a);
  b(ii) = val;
  a(index)= intmax; % replace val stored at a(index) with a very large value
end                 % so that next time the minimum will be the minimum of the remaining keys
disp(b);            % absent of val and smaller than intmax
```

(c) Code fragment `Code3` will not work either. Conside what happens for $ii = 1$ and its sixth line with the `min`. The `min(a(1:len))` will correctly return [1 3] and the first and third keys will get swapped thus changing $a$ into

$$[1 \quad 5 \quad 6 \quad 2 \quad 3 \quad 4]$$

But next time `min(a(2:len))` will attempt to find the minimum of [5 6 2 3 4] and will return [2 3]. The index is that of the subvector not of the original $a$. The position of key value 2 in $a$ is 4 not 3. This will mess up $a$!

```
%Code 3:
 a= [ 6 5 1 2 3 4];
 len =length(a);
 disp(a);
 for ii=1:len-1                    % positions 1... ii-1 contains the ii-1 smallest keys
   [val index] = min(a(ii:len));   % find min of remaining keys in positions ii ... len
   if (index~= ii);  a(index)=a(ii); a(ii)=val; end  % swap a(index) and a(i) if not the same key
 end
 disp(a);
```

The correct way is to adjust `index` appropriately. If `index` is the position of the minimum key in $ii \ldots len$, then `index+ii-1` is its position in the original array $1 \ldots len$.

```
         [ 1  5  6  2  3  4]

The minimum of a(2:len) is the minimum of
             [ 5  6  2  3  4]    which is [2 3] i.e. val =2 and index= 3
But what is the index of the key with val= 2 in the vector
         [ 1  5  6  2  3  4]
Definitely it's not 3 (the value of the key at index 3 is  6!
To determine that index we add to index(=3) the number of keys before 5 i.e. ii-1.
Thus val=2 has index   index+ii-1 = 3 +2 -1 = 4 in
         [ 1  5  6  2  3  4]
```

Then the corrected code for the algorithm known as `SelectionSort` will look like.

```
a= [ 6 5 1 2 3 4];
len =length(a);
disp(a);
for ii=1:len-1
  [val index] = min(a(ii:len));
  if ((index+ii-1)~= ii);  a(index+ii-1)=a(ii); a(ii)=val; end
end
disp(a);
```

(Note that a sorting algorithm rids itself of those lines in the code above that print $a$.) The textbook, starting from page 216, has a in-place version of `SelectionSort` that does not use `min`. Note that the `min` function used in our example generates a subvector through the `a(ii:len)` line. This makes the version above really not in-place! By the way, the algorithm is known as `SelectionSort` because in a given iteration the key **selected** to fill the $i$ position of the output (left-to-right) is the $i$-th smallest key. The 1-st smallest key is known as `minimum` and the $n$-th smallest key of $n$ keys is known as the maximum. The maximum can also be picked to fill the $n - i + 1$ position of the output as well!