SOLUTIONS & Grading Guide

Problem 1 [10 points] *General Concepts*

Answer the following questions. Be concise and clear.

1a [1 point] Who is the course administrator for this course? Hint: Not *DIS*. Not *Schwartz*. And certainly not *Ira*.

Kelly Patwell

1b [1 point] What is the fundamental data type in MATLAB? Hint: Begins with letter "a" and isn't *MATRIX*.

array

1c [2 points] What is a control structure? Give one brief example in MATLAB.

(1) statement (or group of statements) that alters the flow of execution
(1) `while`, `for`, or example of using the structures

1d [2 points] When is it appropriate to number/label one or more steps in an algorithm?

(2) when the step(s) must be referred to by another step

1e [2 points] Distinguish between the *tokens* and *statements* of a programming language.

(1) tokens: words in language
(1) statements: instructions composed of those words (need to say "action" or "instruction")

1f [2 points] Distinguish between the *syntax* and *semantics* of a programming language.

(1) syntax: grammar, structure, spelling of language elements
(1) semantics: meaning of the language elements

Problem 2 [10 points] *MATLAB Short-Answer*

Fill in the boxes for the following MATLAB sessions. Consider each problem as if it were a fresh MATLAB session.

2a [1 point]

```
>> mod(2.1723,5.231)
ans =
```

2.1723

2b [1 point]

```
>> x = 1; y = x; x = 2; y
ans =
```

1

2c [2 points]

```
>> ~xor(1 && (0 | 1), ~(xor(1,0)))
ans =
```

0

2d [2 points]

```
>> sqrt(4) || input('Enter value: ')
```

ans =
1

2e [4 points]

```
>> x = 0; for ii=-3:-1, for jj=ii:-ii:2, x = x+ii+jj; end; end; x + jj
```

ans =
-15

Problem 3 [25 points] *Selection, Repetition (for-loops)*

Problem: Write an M-File script that determines how random MATLAB's **`rand`** function really is. To do so, the program conducts a user-input number of tests in which MATLAB generates bits. Use **`readInt`** for the input. Refer to the second page of this exam for a reminder of the code. Your program will report the amounts of each bit generated in terms of percents and the success or failure of the study. If the difference in amounts is over 5%, **`rand`** failed expectations. For full credit, do *not* use arrays (except for loop indices, scalars, and output strings) for your solution.

Example session:

```
Enter number of tests between 1 and 7000 (inclusive): 100
Testing rand with 100 tests.
0s: 51%
1s: 49%
Result: success!
```

```
tests = readInt(1,7000,'Number of tests: ');
MINDIFF = 5; % difference between 0s and 1s
count0 = 0; % 0 bit counted so far
count1 = 0; % 1 bit counted so far

% generate 1:tests number of bits and count each:
for t=1:tests

  bit = floor(rand*2); % OK if used ROUND, but should be avoided (training for Java)

  if bit % bit == 1
    count1 = count1 + 1;
  else   % bit == 0
    count0 = count0 + 1;
  end

end

% report number of bits and whether or not they are within MINDIFF:
disp(['0s: ',num2str(100*count0/tests),'%'])
disp(['1s: ',num2str(100*count1/tests),'%'])

if (100/tests)*abs(count0-count1) <= MINDIFF
  disp('Result: success!');
else
  disp('Result: failure!');
end
```

```
1: readInt syntax
3: t=1:tests
4: finding random bit
6: choose correct count to increments
3: disp correct
3: results displace
1: style
1: abs
3: syntax (for, if, "0s 1s", == vs =, fprintf vs disp)
```

Problem 4 [55 points] *Code Analysis, Software Design*

Problem: You will write an algorithm (4a) and code (4b) for a script called **`secondmax`** that reports the highest and second-highest grades from user input. The user enters one grade at a time.

Input specifications:

- The lowest possible grade is zero. The highest possible grade is 100.
- The program must use **`readInt`** to process input. Do not allow the user to enter any grade higher than the highest possible grade.
- If the user enters **`-1`**, the program must stop processing grades. Do not allow the user to enter any grade lower than this stopping value.

Output specifications:

- If legal values were entered, the program reports the highest and second highest grades.
- If the user enters all the same values, the program alerts the user that there is no second highest grade.
- If the user enters an insufficient number of grades, the program reports how many grades are missing.

Example session:

```
Welcome to SECONDMAX!
Enter a grade:  10
Enter a grade:  20
Enter a grade:  -1
Maximum: 20
Next Maximum: 10
```

4a [15 points] Write an algorithm for **`secondmax`**. We are looking for correctness, pseudocode, generalization (use of variables when possible), and style (clarity, indentation, choice of wording).

```
initialize variables
   current first <-- -1
   current second <-- -1
   current count <-- 0
get first input for grade

if grade is valid (in bounds)
   increment count of grades
   if grade bigger than current first, swap:
      second <-- first
      first <-- current grade
   else if grade bigger than current second and not first
      second <-- current grade
   get next grade
repeat

report results:
   if no grades (count is zero), report no grades
   else if only one grade (count is one), report one grade
   else if second unchanged from initial value and count > 1
      report only constant grades entered
   else report first and second
```

```
8: correct
   (1: bound, 1: 1st max, 12: 2nd max, 1: -1 or STOP, 2: error mesg, 1: other)
3: pseudocode
3: style
1: general
```

4b [40 points] *Selection, Repetition (while-loops)*

Fill in the blanks and box in the comments and code, below, to complete the program. For full credit, use a **`while`** loop, no arrays (except for scalars and output strings), and only the variables we have supplied.

```
  LOW = 0 ;          % lowest valid grade
  HIGH = 100 ;       % highest valid grade
  STOP = -1 ;        % stopping value
  first  = STOP ;    % highest max grade so far
  second = STOP ;    % second highest max grade so far
  grade  = readInt(STOP,HIGH,'Enter a grade: '); % initial grade
  count = 0;         % number of legal grades entered so far

disp('Welcome to SECONDMAX!');
% process grades until user enters STOP (grade within bounds)
while grade <= HIGH && grade >= LOW % or use STOP

  count = count + 1; % entered grade was legal
  % update first and second grades:
  if grade > first
    second = first;
    first = grade;
  elseif grade > second & grade ~= first
    second = grade;
  end
  grade = readInt(STOP,HIGH,'Enter a grade: ');

end

% report if no grades, constant grades, or first/second:
if count == 0
  disp('No grades entered!');
elseif count == 1;
  disp('Only one valid grade was entered.');
elseif second == STOP & count > 1
  disp('Only constant grades were entered.');
else
  disp(['Maximum: ',num2str(first)]);
  disp(['Next Maximum: ',num2str(second)]);
end
```

NOTE: there are other ways to solve this!

```
Grading:

2: comments                              2: no grades report
1: each "small" blank                    2: 1 grade report
3: input grade blank                     4: constant grade report
1: used WHILE                            2: reports first, second
3: WHILE condition (LOW/HIGH or STOP)    1: used readInt
3: accounted for # of entered grades     1: good readInt
5: updated FIRST                         1: used assigned variables
5: updated SECOND                        1: used constants
2: gets next grade                       1: meaningful variable names
                                         -5 for unnecessary/extra code
```