Information:
Name (clearly print last, first, middle): _____________________________________________________________________
Net ID: _____________________ Course (100M/121): _____________________
CU ID: _____________________
I have followed the rules of academic integrity on this exam (sign): ______________________________________________
Circle your lab section time/place:

<table>
<thead>
<tr>
<th>12:20</th>
<th>1:25</th>
<th>2:30</th>
<th>3:35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue</td>
<td>Upson B7 Blue Room</td>
<td>Red Room Blue Room</td>
<td>Blue Room</td>
</tr>
<tr>
<td>Wed</td>
<td>Upson B7 Blue Room</td>
<td>Red Room Blue Room</td>
<td>Blue Room</td>
</tr>
</tbody>
</table>

We distribute graded exams in lab section. Uncollected exams will be available in Carpenter B101. See Syllabus for details.

Instructions:
Failure to follow any instruction may result in a point deduction on your exam:
• Turn off all cell phones, beepers, pagers, and any other devices that will interrupt the exam.
• Remove all calculators, reference sheets, or any other material. This test is closed book.
• Fill out the information at the top of this exam.
• Wait for us to announce when the exam begins. Skim the entire exam before starting any of the problems.
• Read each problem completely before starting it.
• Write your solutions directly on the test using blue/black pen or pencil. Clearly indicate which problem you are solving.
  You may write on the back of each sheet. If you need scrap paper, ask a proctor.
• Provide only one statement, expression, value, or comment per blank!
• Do not alter, add, or remove any code that surrounds blanks and boxes.
• Do not supply multiple answers. If you do so, we will choose which one to grade.
• Follow good style! When possible, keep solutions general, avoid redundant code, use descriptive variables, use named constants, indent substructures, avoid breaking out of loops, and maintain other tenets of programming philosophy.
• Comment each control structure, major variable, and function (if used), briefly.
• You have 90 minutes. Budget about one minute per point.
• Do not work on bonus problems (if there are any) until you have thoroughly proofread all required (core-point) problems!
• Try to figure out problems yourself before raising your hand so that we can avoid disturbing people in cramped rooms.

Core Points:

1. _______ (10 points) _________
2. _______ (10 points) _________
3. _______ (25 points) _________
4a. _______ (15 points) _________
4b. _______ (40 points) _________
Total: _______ / (100 points) _________
Reminders

\begin{align*}
\text{abs}(x) & \quad \% \text{ absolute value of } x \\
\text{disp} & \quad \% \text{ displays array of strings} \\
\text{help} & \quad \% \text{ finds help :-)} \\
\text{input(s)} & \quad \% \text{ prompts with prompt } S \text{ the user to enter an expression} \\
\text{floor} & \quad \% \text{ returns integer portion of a number} \\
\text{mod}(x,y) & \quad \% \text{ returns the remainder of } X./Y \\
\text{num2str} & \quad \% \text{ converts a numerical value to a string} \\
\text{rand} & \quad \% \text{ returns random value between 0 and 1, non-inclusive}
\end{align*}

\begin{verbatim}
function value = readInt(min,max,prompt) \% same as shown in lab and lecture 
\% READINT Get a user-input integer 
\% readInt is cleaner version of MATLAB’s INPUT function. 
\% You can essentially bang on the keyboard without causing a problem.

\% Ensure that the prompt is a string:
\% if \texttt{\textasciitilde ischar(prompt)}
\% \hspace{1cm} \text{error(‘Your prompt should be a string!’);}
\% \end

\% Get initial input from the user:
value = str2double(input(prompt,’s’));

\% Ensure that the input it legal:
while \texttt{\textasciitilde isreal(value)} & \hspace{1cm} \% no complex numbers
\texttt{isinf(value)} \hspace{1cm} \% no infinite values
\texttt{isnan(value)} \hspace{1cm} \% no NaNs
\texttt{\textasciitilde isnumeric(value)} \hspace{1cm} \% no non-numerical values
\texttt{floor(value)\textasciitilde ceil(value)} \hspace{1cm} \% no decimal/fraction values
\texttt{value > max} \hspace{1cm} % lower bound of integer
\texttt{value < min} \hspace{1cm} % upper bound of integer

\% Reprompt the user and get revised input:
\% disp(‘That value is not legal! Please re-enter: ’);
\% value = str2double(input(prompt,’s’));

\end
\end{verbatim}
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.

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

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

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

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

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

2b  [1 point]
>> x = 1; y = x; x = 2; y
ans =

2c  [2 points]
>> ~xor(1 && (0 | 1), ~(xor(1,0)))
ans =

2d  [2 points]
>> sqrt(4) || input('Enter value: ')

2e  [4 points]
>> x = 0; for ii=-3:-1, for jj=ii:-ii:2, x = x+ii+jj; end; end; x + jj
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!
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 unique 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: 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).
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 \texttt{while} loop, no arrays (except for scalars and output strings), and only the variables we have supplied.

% Initialize variables:

\begin{verbatim}
LOW = 0 ;  % ___________________________

HIGH = 100 ;  % ___________________________

STOP = ______ ;  % stopping value

first = _________ ;  % highest max grade so far

second = _________ ;  % second highest max grade so far

grade = _____________________________________________________  % initial grade

count = __________ ;  % number of valid grades so far

% Rest of program:
\end{verbatim}