Review of topics for Test 1

- **Announcements/Reminders:**
  - Assignment 1b due Nov 4th 11:59pm
  - Test 1 this Wednesday
  - Assignment 1a regrade till this Saturday (8th)
Characters & strings

- We have used strings already:
  - `n = input('Next number: ')`
  - `fprintf('Answer is %d', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- `'CS1112 rocks!'` is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.
- Can have 2-d array of characters as well

\[
\begin{array}{cccccc}
'C' & 'S' & '1' & '1' & '3' & '2' \\
'r' & 'o' & 'c' & 'k' & 's' & '!' \\
\end{array}
\]

\[
\begin{array}{cccccc}
'C' & 'S' & '1' & '1' & '3' & '2' \\
'r' & 'o' & 'c' & 'k' & 's' & '!' \\
\end{array}
\]

\[
2 \times 6 \text{ matrix}
\]
Matlab types: `char`, `double`, `uint8`, `logical`

There is not a type “string”! What we call a string is a 1-d array of chars

\[
a = \begin{bmatrix} C \ S \ 1 \end{bmatrix}
\]

\(a\) is a 1-d array with type \texttt{char} components. We call \(a\) a “string” or “char array”

\[
b = [3 \ 9]
\]

\(b\) is a 1-d array with type \texttt{double} components. \texttt{double} is the default type for numbers in Matlab. We call \(b\) a “numeric array”

\[
d = \text{rand} > .5
\]

\(d\) is a scalar of the type \texttt{logical}. We call \(d\) a “boolean value”
Single quotes enclose strings in Matlab

Anything enclosed in single quotes is a string (even if it looks like something else)

- ‘100’ is a character array (string) of length 3
- 100 is a numeric value
- ‘pi’ is a character array of length 2
- pi is the built-in constant 3.1416…
- ‘x’ is a character (vector of length 1)
- x may be a variable name in your program
### Vectors
- **Assignment**
  
  \[
  v = [7 \ 0 \ 5];
  \]
- **Indexing**
  
  \[
  x = v(3); \quad \% \ x \ is \ 5 \\
  v(1) = 1; \quad \% \ v \ is \ [1 \ 0 \ 5] \\
  w = v(2:3); \quad \% \ w \ is \ [0 \ 5]
  \]
- **: notation**
  
  \[
  v = 2:5; \quad \% \ v \ is \ [2 \ 3 \ 4 \ 5]
  \]
- **Appending**
  
  \[
  v = [7 \ 0 \ 5]; \\
  v(4) = 2; \quad \% \ v \ is \ [7 \ 0 \ 5 \ 2]
  \]
- **Concatenation**
  
  \[
  v = [v \ [4 \ 6]]; \\
  \quad \% \ v \ is \ [7 \ 0 \ 5 \ 2 \ 4 \ 6]
  \]

### Strings
- **Assignment**
  
  \[
  s = ‘hello’;
  \]
- **Indexing**
  
  \[
  c = s(2); \quad \% \ c \ is \ ‘e’ \\
  s(1) = ‘J’; \quad \% \ s \ is \ ‘Jello’ \\
  t = s(2:4); \quad \% \ t \ is \ ‘ell’
  \]
- **: notation**
  
  \[
  s = ‘a’:‘g’; \quad \% \ s \ is \ ‘abcdefg’
  \]
- **Appending**
  
  \[
  s = ‘duck’; \\
  s(5) = ‘s’; \quad \% \ s \ is \ ‘ducks’
  \]
- **Concatenation**
  
  \[
  s = [s \ ‘quack’]; \\
  \quad \% \ s \ is \ ‘ducks quack’
  \]
Some useful string functions

str = 'Cs 1112';

length(str) % 7
isletter(str) % [1 1 0 0 0 0 0 0]
isspace(str) % [0 0 1 0 0 0 0 0]
lower(str) % 'cs 1112'
upper(str) % 'CS 1112'

ischar(str)
  % Is str a char array? True (1)
strcmp(str(1:2), 'cs')
  % Compare strings str(1:2) & 'cs'. False (0)
strcmp(str(1:3), 'CS')
  % False (0)
Example: capitalize 1\textsuperscript{st} letter

Write a function to capitalize the first letter of each word in a string. Assume that the string has lower case letters and blanks only. (OK to use built-in function \texttt{upper})

function [str, nCaps] = caps(str)
% Post: Capitalize first letter of each word.
% str = partially capitalized string
% nCaps = no. of capital letters
% Pre: str = string with lower case letters & blanks only

look for the spaces
Look For The Spaces

See caps.m
function [str, nCaps] = caps(str)

% Post: Capitalize 1st letter of each word.
% str= partially capitalized string
% nCaps= no. of capital letters
% Pre: str= string with lower case letters and blanks only

nCaps= 0;

for k= 2:length(str)
    if (str(k-1)==" " && isletter(str(k)))
        str(k)= upper(str(k));
        nCaps= nCaps + 1;
    end
end

if (isletter(str(1)))
    str(1)= upper(str(1));
    nCaps= nCaps + 1;
end
# ASCII characters

(American Standard Code for Information Interchange)

<table>
<thead>
<tr>
<th>ascii code</th>
<th>Character</th>
<th>ascii code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>65</td>
<td>‘A’</td>
<td>48</td>
<td>‘0’</td>
</tr>
<tr>
<td>66</td>
<td>‘B’</td>
<td>49</td>
<td>‘1’</td>
</tr>
<tr>
<td>67</td>
<td>‘C’</td>
<td>50</td>
<td>‘2’</td>
</tr>
<tr>
<td>90</td>
<td>‘Z’</td>
<td>57</td>
<td>‘9’</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Character vs ASCII code

\[
\text{str} = 'Age 19' \\
\quad \% \text{a 1-d array of characters} \\
\text{code} = \text{double}(\text{str}) \\
\quad \% \text{convert chars to ascii values} \\
\text{str1} = \text{char}(\text{code}) \\
\quad \% \text{convert ascii values to chars}
\]
Arithmetic and relational ops on characters

- `'c' - 'a'` gives 2
- `'6' - '5'` gives 1
- `letter1='e'; letter2='f';`
- `letter1-letter2` gives -1
- `'c'>'a'` gives true
- `letter1==letter2` gives false

- `'A' + 2` gives 67
- `char('A'+2)` gives 'C'
What is in variable \( g \) (if it gets created)?

\[
d1 = 'Mar 3'; \\
d2 = 'Mar 9'; \\
x1 = d1(5); \\
x2 = d2(5); \\
g = x2 - x1;
\]

A: the character ‘6’
B: the numeric value 6
C: Error in the subtraction operation
D: Error in assigning variables \( x1, x2 \)
E: Some other value or error
What is in variable $g$ (if it gets created)?

d1= 'Mar 13';  d2= 'Mar 29';
x1= d1(5:6);  x2= d2(5:6);
g= x2-x1;

A: the string ‘16’
B: the numeric value 16
C: Error in the subtraction operation
D: Error in assigning variables x1, x2
E: Some other value or error
Example: \texttt{toUpperCase}

Write a function \texttt{toUpperCase(cha)} to convert character \texttt{cha} to upper case if \texttt{cha} is a lower case letter. Return the converted letter. If \texttt{cha} is not a lower case letter, simply return the character \texttt{cha}.

\textbf{Hint:} Think about the distance between a letter and the base letter ‘\texttt{a}’ (or ‘\texttt{A}’). E.g.,

\begin{center}
\begin{tabular}{ccccccccccc}
\texttt{a} & \texttt{b} & \texttt{c} & \texttt{d} & \texttt{e} & \texttt{f} & \texttt{g} & \texttt{h} & \ldots \\
A & B & C & D & E & F & G & H & \ldots \\
\end{tabular}
\end{center}

\texttt{distance} = ‘\texttt{g}’-‘\texttt{a}’ = 6 = ‘\texttt{G}’-‘\texttt{A}’

Of course, do not use Matlab function \texttt{upper}!
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.
up= cha;

cha is lower case if it is between 'a' and 'z'
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.
up = cha;
if ( cha >= 'a' && cha <= 'z' )
    % Find distance of cha from ‘a’
end
end
function up = toUpper(cha)

% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from 'a'
    offset= cha - 'a';

    % Go same distance from 'A'
end
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from 'a'
    offset= cha - 'a';

    % Go same distance from 'A'
    up= char('A' + offset);
end
<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>HTML</th>
<th>Chr</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>HTML</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>#32;</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>#64;</td>
<td>@</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>#33;</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>#65;</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>#34;</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>#66;</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>#35;</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>#67;</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>#36;</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>#68;</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>#37;</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>#69;</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>#38;</td>
<td>&lt;</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>#70;</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>#39;</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>#71;</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>#40;</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>#72;</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>#41;</td>
<td></td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>#73;</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>#42;</td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>#74;</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>#43;</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>#75;</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>#44;</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>#76;</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>#45;</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>#77;</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>#46;</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>#78;</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>#47;</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>#79;</td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>#48;</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>#80;</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>#49;</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>#81;</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>#50;</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>#82;</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>#51;</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>#83;</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>#52;</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>#84;</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>#53;</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>#85;</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>#54;</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>#86;</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>#55;</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>#87;</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td>#56;</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>#88;</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>#57;</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>#89;</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>#58;</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>#90;</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>#59;</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>#91;</td>
<td>[</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>#60;</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>#92;</td>
<td>\</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>#61;</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>#93;</td>
<td>]</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>#62;</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>#94;</td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>#63;</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>#95;</td>
<td>_</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Review

- Script and Function
- If statement
- While loop
- For loop
- Array
- Matrix
- Plot, hold on, legend
The `if` construct

```plaintext
if  boolean expression1
    statements to execute if  expression1  is true
elseif  boolean expression2
    statements to execute if  expression1  is false
    but  expression2  is true
:
else
    statements to execute if all previous conditions
    are false
end
```

Can have any number of elseif branches but at most one else branch
Generating random numbers

- $\text{rand}(m,n)$ gives an $m$-by-$n$ matrix of random values, each in interval $(0,1)$

- Generate a random number in the range $(a,b)$

- Generate a random integer in the range $[a,b]$
Generating random numbers

- \texttt{rand(m,n)} gives an m-by-n matrix of random values, each in interval (0,1)

- Generate a random number in the range (a,b)
  \[ \text{rand} \times (b-a) + a \]

- Generate a random integer in the range [a,b]
  \[ \text{floor} \left( \text{rand} \times (b-a+1) + a \right) \]
  \[ \text{ceil} \left( \text{rand} \times (b-a+1) + a-1 \right) \]
Simulation problem:

- Ann and Bob take turns flipping an unfair coin—twice as likely to be heads than tails.
- In one round, each player flips once.
- Ann gets 1 point if she gets heads; Bob gets 2 points if he gets tails.
- Game ends after the round in which at least one player gets 10 points. Display the final scores.
Simulation problem:  

- Ann and Bob take turns flipping an unfair coin—twice as likely to be heads than tails.
- In one round, each player flips once.
- Ann gets 1 point if she gets heads; Bob gets 2 points if he gets tails.
- Game ends after the round in which at least one player gets 10 points. Display the final scores.

```java
step : pA >= 10 or pB >= -10

- Ann and Bob take turns flipping an unfair coin—twice as likely to be heads than tails.
- In one round, each player flips once.
- Ann gets 1 point if she gets heads; Bob gets 2 points if he gets tails.
- Game ends after the round in which at least one player gets 10 points. Display the final scores.

```
Common loop patterns

Do something \( n \) times

\[
\text{for } k = 1:1:n \\
\quad \% \text{ Do something} \\
\text{end}
\]

Do something an indefinite number of times

\[
\% \text{Initialize loop variables} \\
\text{while } ( \text{not stopping signal} ) \\
\quad \% \text{ Do something} \\
\quad \% \text{ Update loop variables} \\
\text{end}
\]
for loop examples

for k = 2:0.5:3
    k takes on the values 2, 2.5, 3
    disp(k)
end
Non-integer increment is OK

for k = 1:4
    k takes on the values 1, 2, 3, 4
    disp(k)
end
Default increment is 1

for k = 0:-2:-6
    k takes on the values 0, -2, -4, -6
    disp(k)
end
“Increment” may be negative

for k = 0:-2:-7
    k takes on the values 0, -2, -4, -6
    disp(k)
end
Colon expression specifies a bound

for k = 5:2:1
    k takes on the values 5
    disp(k)
end
The set of values for k is the empty set: the loop body won’t execute
for k = 4:6
    disp(k)
    k = 9;
    disp(k)
end

Not a condition (boolean expression) that checks whether k<=6.

It is an expression that specifies values:

4 5 6
Built-in functions for creating/manipulating arrays

- **Creation**
  - zeros, ones, rand
  - linspace

- **Manipulation**
  - length
  - size
Built-in functions for creating/manipulating arrays

- **Creation**
  - zeros, ones, rand
  - linspace

- **Manipulation**
  - length
  - size

\[
\text{zeros} \left(3, 2\right) \\
\text{linspace}\left(\frac{3}{10}, \frac{8}{10}\right) \\
[a_{\text{row}}, a_{\text{col}}] = \text{size}(M) \\
\begin{bmatrix}4 & 2 & 3 & \text{ones}(2,3)\end{bmatrix}
\]
Example

- Write a program fragment that calculates the cumulative sums of a given vector \( v \).
- The cumulative sums should be stored in a vector of the same length as \( v \).

\[ 1, 3, 5, 0 \quad v \]

\[ 1, 4, 9, 9 \quad \text{cumulative sums of } v \]
\[ cSum(k) = cSum(k-1) + V(k) \]

\[ cSum(3) = V(1) + V(2) + V(3) \]
\[ cSum(4) = V(1) + V(2) + V(3) + V(4) \]

\[ cSum(1) = V(1); \]
for \( k = 2 : \text{length}(V) \)
\[ cSum(k) = cSum(k-1) + V(k); \]
end
Function header is the “contract” for how the function will be used (called)

You have this function:

```matlab
function [x, y] = polar2xy(r, theta)
% Convert polar coordinates (r, theta) to
% Cartesian coordinates (x,y). Theta in degrees.
...
```

Code to call the above function:

```matlab
% Convert polar (r1,t1) to Cartesian (x1,y1)
r1 = 1; t1 = 30;
[x1, y1] = polar2xy(r1, t1);
plot(x1, y1, 'b*')
...
```
Given this function:

```matlab
function m = convertLength(ft,in)
    % Convert length from feet (ft) and inches (in) to meters (m).

    % ... 
```

How many proper calls to `convertLength` are shown below?

```matlab
% Given f and n
d = convertLength(f,n);
d = convertLength(f*12+n);
d = convertLength(f+n/12);
x = min(convertLength(f,n), 1);
y = convertLength(pi*(f+n/12)^2);
```

A: 1  B: 2  C: 3  D: 4  E: 5 or 0
Example

- Write a function `evalPoly` to evaluate an $n^{th}$ order polynomial of $x$:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

- Input parameter `coef` has length $n+1$, contains the coefficients of the polynomial

  - `coef(1)` is the coefficient for the term $x^0$

- Input parameter `x`

- Return the value of the polynomial evaluated at $x$

- No Matlab predefined function other than `length`
$c_{0}c_{f}$

$\begin{array}{c|cccc}
1 & 2 & 3 & 4 \\
\end{array}$

$c_{1}x^{0} + c_{2}x^{1} + c_{3}x^{2} + c_{4}x^{3}$
function val = evalPoly ( coef , x )
% val is polynomial evaluated at x
% coef is a vector where coef (1) is for term x^0
% xpow = 1;
val = coef (1)
for k = 2 : length (coef )
    val = val + coef ( k )* x^ ( k-1 ) ;
    % xpow = xpow * x ;
    % val = val + coef ( k )* xpow ;
end
Other notes for the test (course)

- Read questions/instructions carefully
- Use Matlab syntax
- Do not use `break, continue`
- Do not use `randi`, instead use `rand`
- Many students make “index out-of-bounds” error
Other notes for the test (course)

- Read questions/instructions carefully
- Use Matlab syntax
- Do not use `break`, `continue`
- Do not use `randi`, instead use `rand`
- Many students make “index out-of-bounds” error

```matlab
%% vector v
for k = 1 : length (v) - 1
    \( x(k) = v(k) + v(k + 1) \)
end
```
Review

- How to create script
- Logic operator
  - ==
  - &&
  - ||
  - ~=
  - >=
  - >=
  - <=
- Built in function
  - rand
  - fix, floor, ceil
  - disp, fprintf
Review exercise

suppose $a = 1$, $b = 3$

if $a > 1$ && $b + a > 2$
    $a = 3$
else
    $a = 4$
end

Question: $a + b = ?$
Review exercise

Suppose \( a = 1, b = 3 \)

\[
\text{if } a > 1 \land b + a > 2 \\
a = 3; \\
\text{elseif } a > -1 \\
a = 4; \\
\text{else} \\
a = 5; \\
\text{end}
\]

**Question:** \( a + b = ? \)
How to use function and scripts

- function out_var=my function(inp_var)
  % Comments that explain what the function does
  computations
  out_var=desired result;

- End

- Exercise 1:
  - Try to create a function that adds two numbers
  - The function is called addtwo.m
  - Input: x, y
  - Output: z
Exercise 2:

```matlab
for x = 1:2:15
    disp(x)
end
```
Convert the For loop to While loop

for x = 1:2:15
    disp(x)
end

% For loop with counter increment
x = 1;  % STEP 1
while(x <= 15)  % STEP 2
    disp(x)
    x = x + 2;  % STEP 3
end
For loop

- Duplicate each element in an array
- Input: Array [1,2,3,4]
- Output: Array [1,1,2,2,3,3,4,4]
V = 1:5; % sample vector

%%% method 1
for j = 1:length(V)
    V2((2*j-1):(2*j)) = V(j);
end
disp(V2)

%%% method 2
for j = 1:(2*length(V))
    V2(j) = V(ceil(j/2));
end
disp(V2)

%%% method 3
counter = 0.5;
for j = 1:(2*length(V))
    V2(j) = V(ceil(counter));
    counter = counter + 0.5;
end
disp(V2)
Challenge Question: Nested Loops

- Printing Out rows of Stars
- Ask the user for how many rows

* 
** 
*** 
**** 
***** 
****** 
******* 
******** 
...

...
Challenge Question: Nested Loops

```plaintext
rows = input('How many rows do you want?');
for R = 1:rows
    for s = 1:R
        fprintf('*');
    end
    fprintf('
');
end
```
Array display

- Create 3 loops that display the following three arrays:

  
  0  4  8  12  16  20  24  28
  69  68  67  66  65  64  63  62
  1.4 1.1 0.8 0.5 0.2 -0.1 -0.4 -0.7

- Make sure you display the array in the same way shown above.
  - Hint: Use fprintf, %d, %f, \n
for i = 0:4:28
    fprintf('%6d',i)
end
fprintf('
')

for i = 69:-1:62
    fprintf('%6d',i)
end
fprintf('
')

for i = 1.4:-0.3:-0.7
    fprintf('%6.1f',i)
end
fprintf('
')
Array display

- Create 3 loops that display the following three arrays:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>69</td>
<td>67</td>
<td>66</td>
<td>65</td>
<td>64</td>
<td>63</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>1.1</td>
<td>0.8</td>
<td>0.5</td>
<td>0.2</td>
<td>-0.1</td>
<td>-0.4</td>
<td>-0.7</td>
<td></td>
</tr>
</tbody>
</table>

- Can you use one loop?
Nested Loops

- Printing Out rows of Stars
- Ask the user for how many rows

* 
** 
*** 
**** 
***** 
****** 
******* 
******** 
*********

...
Nested Loops

```python
rows = input('How many rows do you want?');
for R = 1:rows
    for s = 1:R
        fprintf('*');
    end
    fprintf('
');
end
```