- Previous Lecture:
  - Iteration using **while**

- Today’s Lecture:
  - Nested loops
  - Developing algorithms

- Announcements:
  - Read *Insight* §3.2 before discussion, in the lab this week
  - Project 2 Parts A & B due Thurs 2/14 at 11pm
  - We do not use **break** in this course
  - Make use of Piazza, office hrs, and consulting hrs
What is the last line of output?

```matlab
x = 1;
disp(x)
y = x;
while y==x && x<=4 && y<=4
    x = 2*x;
    disp(x)
end
```

A: 1  B: 2  C: 4  D: 8
Example: Nested Stars
Example: Nested Stars

Draw a black square
- Bigger than the biggest star (at least 2 times radius of star)
- Center at (0,0)

Draw a sequence of stars
- Stars alternate in color
- Stars get smaller
  - radius $r=1$ to start
- $1^{st}$ star smaller than the sqr
- When to stop?
  - when $r$ is small
nestedStars.m
Knowing how to draw

How difficult is it to draw
Pattern for doing something $n$ times

$$n = _____$$

for $k = 1:n$

\[
\% \text{ code to do} \\
\% \text{ that something}
\]

end
x = 0; y = 0; % figure centered at (0,0)

s = 2.1; % side length of square
DrawRect(x-s/2,y-s/2,s,s,'k')

r = 1; k = 1;
while r > 0.1 % r still big
    % draw a star
    if rem(k,2)==1 % odd number
        DrawStar(x,y,r,'m') % magenta
    else
        DrawStar(x,y,r,'y') % yellow
    end
    % reduce r
    r = r/1.2;
k = k + 1;
end
for c = 0:2:8

x = c; y = c; % figure centered at (c,c)

s = 2.1; % side length of square
DrawRect(x-s/2,y-s/2,s,s,'k')

r = 1; k = 1;
while r > 0.1 % r still big
    % draw a star
    if rem(k,2) == 1 % odd number
        DrawStar(x,y,r,'m') %magenta
    else
        DrawStar(x,y,r,'y') %yellow
    end
    % reduce r
    r = r/1.2;
k = k + 1;
end
Pattern for doing something $n$ times

\[ n = \_\_\_\_\_ \]
\[ \text{for } k = 1:n \]
\[ \% \text{ code to do} \]
\[ \% \text{ that something} \]
\[ \text{end} \]
Example: Are they prime?

- Given integers $a$ and $b$, write a program that lists all the prime numbers in the range $[a, b]$.
- Assume $a > 1$, $b > 1$ and $a < b$. 
Example: Are they prime?
Subproblem: Is it prime?

- Given integers a and b, write a program that lists all the prime numbers in the range \([a, b]\).
- Assume \(a > 1\), \(b > 1\) and \(a < b\).
- Write a program fragment to determine whether a given integer \(n\) is prime, \(n > 1\).
- Reminder: \(\text{rem}(x, y)\) returns the remainder of \(x\) divided by \(y\).
Example: Are they prime?  
Subproblem: Is it prime?

- Given integers \( a \) and \( b \), write a program that lists all the prime numbers in the range \([a, b]\).
- Assume \( a > 1 \), \( b > 1 \) and \( a < b \).
- Write a program fragment to determine whether a given integer \( n \) is prime, \( n > 1 \).
- Reminder: \( \text{rem}(x, y) \) returns the remainder of \( x \) divided by \( y \).
Start:
  divisor = 2
Repeat:
  rem (n, divisor)
  divisor = divisor + 1
End:
  rem (n, divisor) = 0
  divisor < n?

\[ \text{divisor} = 2; \]
\[ \text{while } (\text{rem} (n, \text{divisor}) \neq 0) \]
\[ \quad \text{divisor} = \text{divisor} + 1; \]
\[ \text{end} \]
\[ \text{if } (\text{divisor} = = n) \]
\[ \quad \text{disp ('prime')} \]
\[ \text{else} \]
\[ \quad \text{disp ('composite')} \]
\[ \text{end} \]
%Given n, display whether it is prime

divisor = 2;
while (rem(n, divisor) ~= 0)
    divisor = divisor + 1;
end
if (divisor == n)
    fprintf('%d is prime\n', n)
else
    fprintf('%d is composite\n', n)
end
for n = a:b

%Given n, display whether it is prime
divisor = 2;
while (rem(n,divisor) ~= 0)
    divisor = divisor + 1;
end
if (divisor == n)
    fprintf('%d is prime\n', n)
else
    fprintf('%d is composite\n', n)
end
end
Example: Times Table

Write a script to print a times table for a specified range.

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
</tr>
</tbody>
</table>
Developing the algorithm for the times table

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
</tr>
</tbody>
</table>
Developing the algorithm for the times table

- **Look for patterns**
  - Each entry is $\text{row#} \times \text{col#}$
  - Row#, col# increase regularly
- $\Rightarrow \text{Loop}!!!$

- **What kind of loop?**
  - **for-loop**—since the range of the headings will be specified and increment regularly
  - for each row#, get the products with all the col#s. Then go to next row# and get products with all col#s, …
- $\Rightarrow \text{Nested loops!}$

- **Details:** what will be the print format? Don’t forget to start new lines. Also need initial input to specify the range.
disp('Show the times table for specified range')
lo= input('What is the lower bound? ');
hi= input('What is the upper bound? ');
Rational approximation of $\pi$

- $\pi = 3.141592653589793…$
- Can be closely approximated by fractions, e.g., $\pi \approx 22/7$
- Rational number: a quotient of two integers
- Approximate $\pi$ as $p/q$ where $p$ and $q$ are positive integers $\leq M$
- Start with a straightforward solution:
  - Get $M$ from user
  - Calculate quotient $p/q$ for all combinations of $p$ and $q$
  - Pick best quotient $\rightarrow$ smallest error