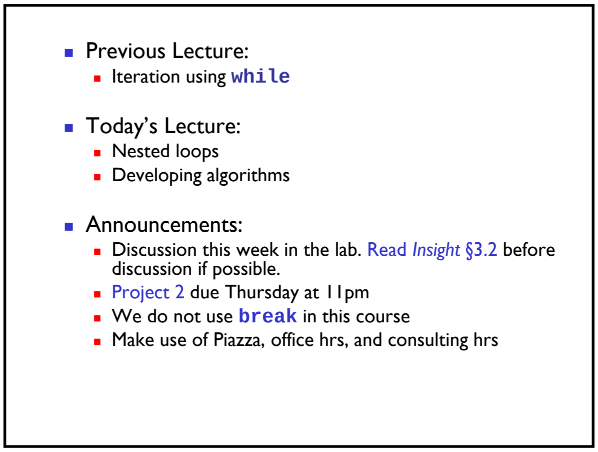
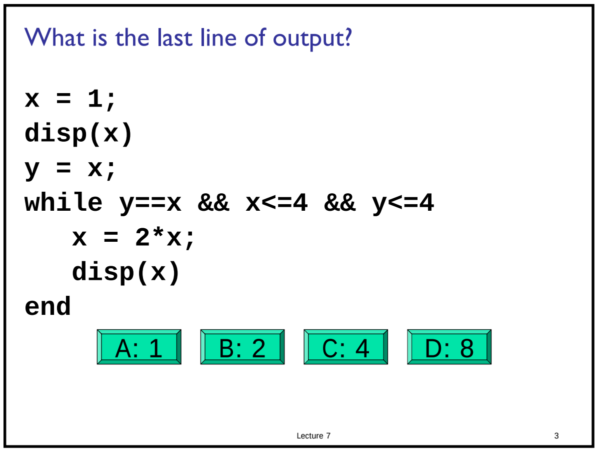


- Previous Lecture:
  - Iteration using `while`
- Today's Lecture:
  - Nested loops
  - Developing algorithms
- Announcements:
  - Discussion this week in the lab. Read *Insight* §3.2 before discussion if possible.
  - Project 2 due Thursday at 11pm
  - We do not use `break` in this course
  - Make use of Piazza, office hrs, and consulting hrs

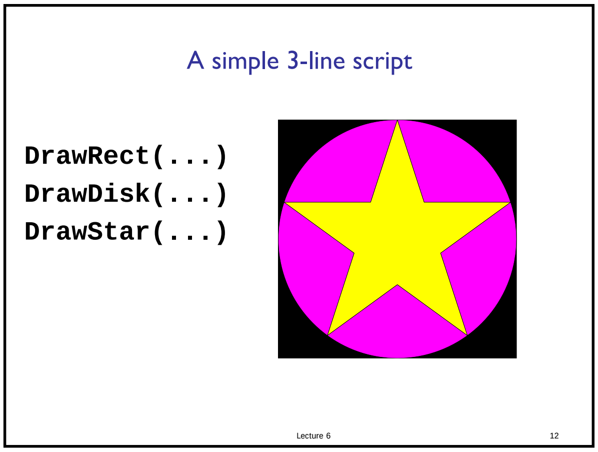


What is the last line of output?

```
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

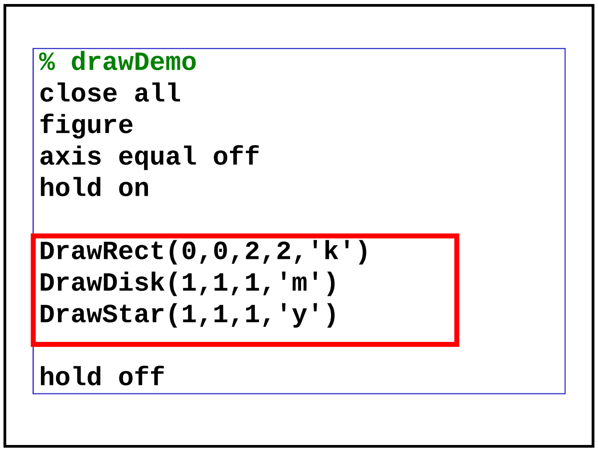
Lecture 7 3



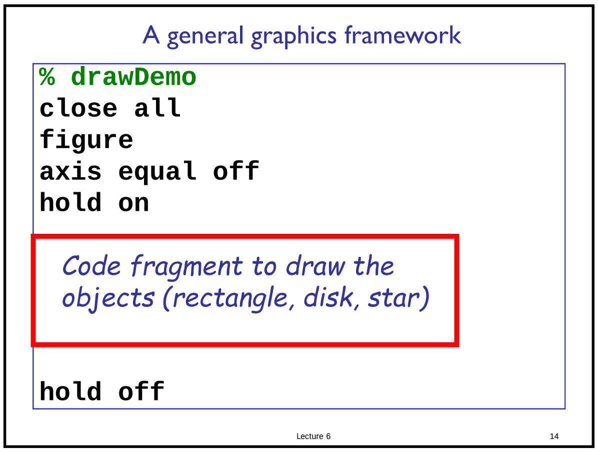
A simple 3-line script

```
DrawRect(...)
DrawDisk(...)
DrawStar(...)
```

Lecture 6 12



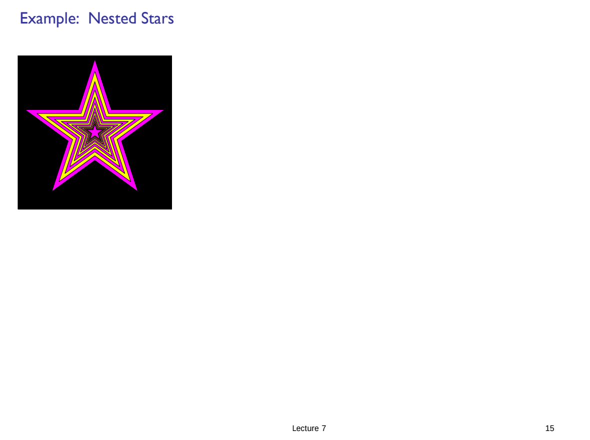
```
% drawDemo
close all
figure
axis equal off
hold on

DrawRect(0,0,2,2,'k')
DrawDisk(1,1,1,'m')
DrawStar(1,1,1,'y')

hold off
```



A general graphics framework

```
% drawDemo
close all
figure
axis equal off
hold on

  Code fragment to draw the
  objects (rectangle, disk, star)

hold off
```

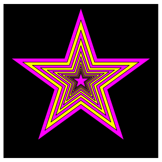
Lecture 6 14



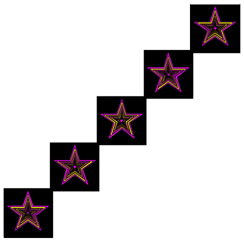
Example: Nested Stars

Lecture 7 15

Knowing how to draw



How difficult is it to draw



Pattern for doing something *n* times

```
n= _____
for k= 1:n

   % code to do
   % that something

end
```

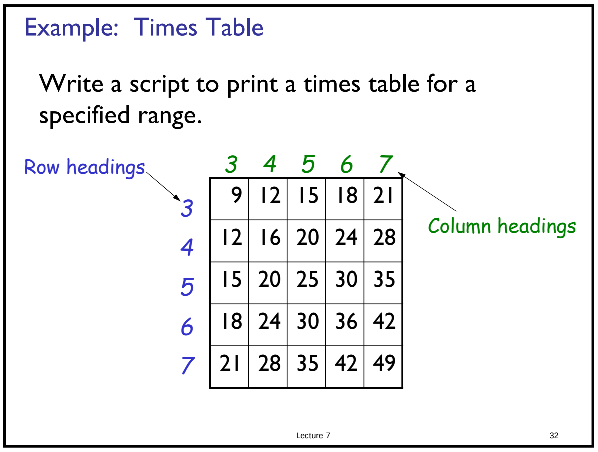
```
x= 0; y= 0; % figure centered at (0,0)

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
```

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: rem(x,y) returns the remainder of x divided by y.



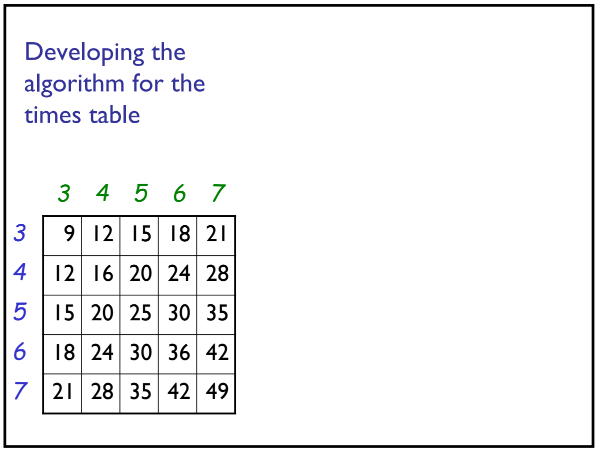
Example: Times Table

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

Row headings

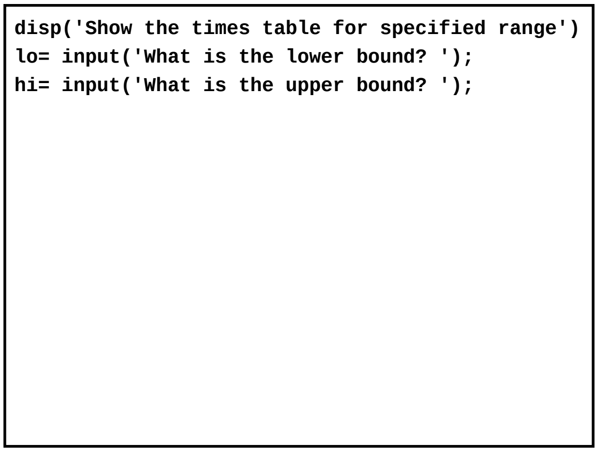
Column headings

	3	4	5	6	7
3	9	12	15	18	21
4	12	16	20	24	28
5	15	20	25	30	35
6	18	24	30	36	42
7	21	28	35	42	49

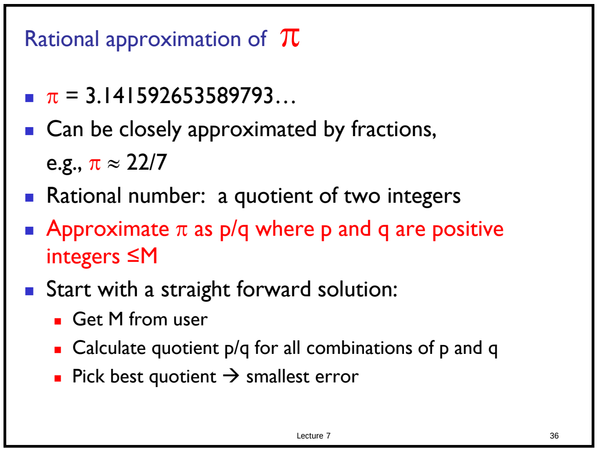


Developing the algorithm for the times table

	3	4	5	6	7
3	9	12	15	18	21
4	12	16	20	24	28
5	15	20	25	30	35
6	18	24	30	36	42
7	21	28	35	42	49

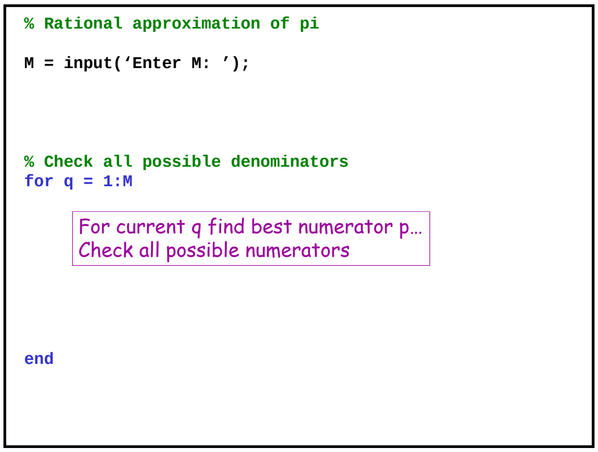


```
disp('Show the times table for specified range')
lo= input('What is the lower bound? ');
hi= input('What is the upper bound? ');
```

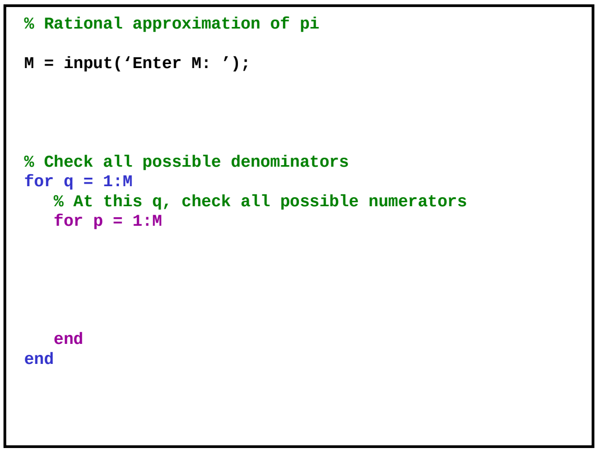


Rational approximation of π

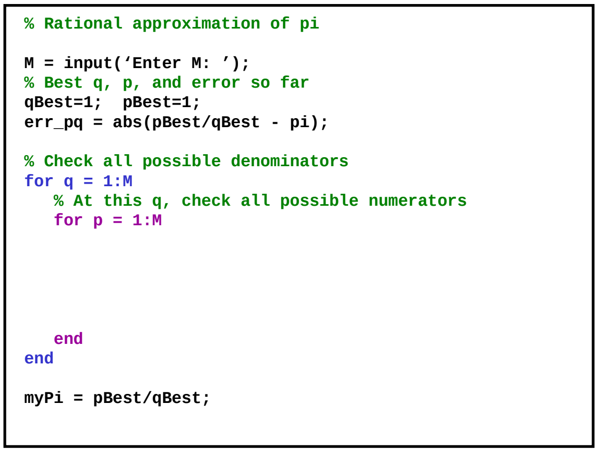
- π = 3.141592653589793…
- Can be closely approximated by fractions, e.g., π ≈ 22/7
- Rational number: a quotient of two integers
- Approximate π as p/q where p and q are positive integers ≤M
- Start with a straight forward solution:
  - Get M from user
  - Calculate quotient p/q for all combinations of p and q
  - Pick best quotient → smallest error



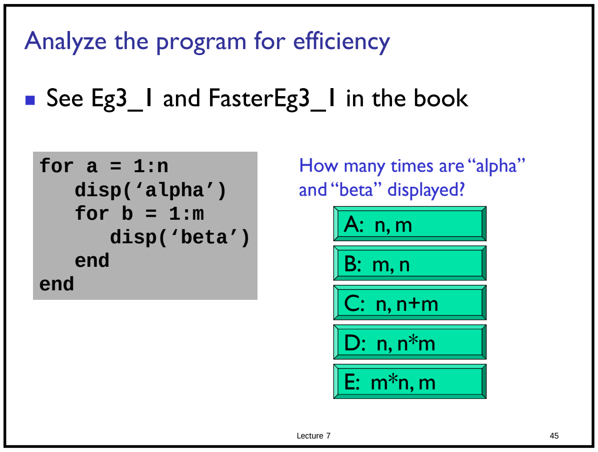
```
% Rational approximation of pi

M = input('Enter M: ');



% Check all possible denominators
for q = 1:M

     For current q find best numerator p…
     Check all possible numerators




end
```



```
% Rational approximation of pi

M = input('Enter M: ');



% Check all possible denominators
for q = 1:M
   % At this q, check all possible numerators
   for p = 1:M





   end
end
```



```
% Rational approximation of pi

M = input('Enter M: ');
% Best q, p, and error so far
qBest=1;  pBest=1;
err_pq = abs(pBest/qBest - pi);

% Check all possible denominators
for q = 1:M
   % At this q, check all possible numerators
   for p = 1:M





   end
end

myPi = pBest/qBest;
```



Analyze the program for efficiency

- See Eg3_1 and FasterEg3_1 in the book

```
for a = 1:n
   disp('alpha')
   for b = 1:m
      disp('beta')
   end
end
```

How many times are "alpha" and "beta" displayed?

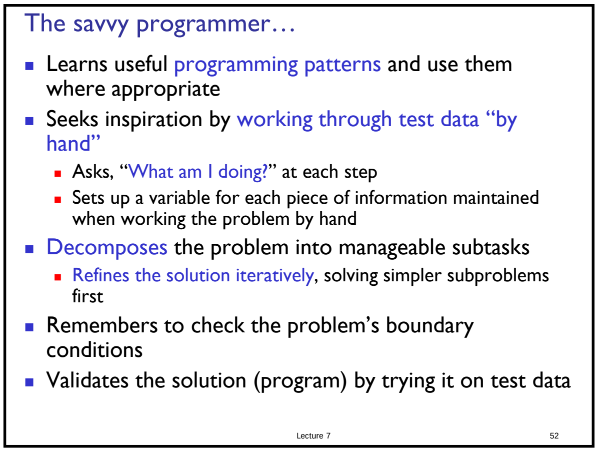
A: n, m

B: m, n

C: n, n+m

D: n, n*m

E: m*n, m



The savvy programmer…

- Learns useful programming patterns and use them where appropriate
- Seeks inspiration by working through test data "by hand"
  - Asks, "What am I doing?" at each step
  - Sets up a variable for each piece of information maintained when working the problem by hand
- Decomposes the problem into manageable subtasks
  - Refines the solution iteratively, solving simpler subproblems first
- Remembers to check the problem's boundary conditions
- Validates the solution (program) by trying it on test data