## Question 1: (10 points)

**Part (a):** (3 points)

What are the final values of variables x and y?

```
x= 4;
y= 8;
x= y;
y= x;
```

x $\boxed{8}$

y $\boxed{8}$

**Part (b):** (3 points)

What are the final values of variables x and y?

```
x= 4;  y= 8;
if x<5
   x= 1;
elseif x<2 && y<10
   y= 11;
end
```

x $\boxed{1}$

y $\boxed{8}$

**Part (c):** (4 points)

Assume that variables **a**, **b**, and **c** store real values and **a**<**b**. Write an expression that evaluates to *true* if **c** is in the open interval (**a**,**b**) or the interval (**a**,**b**) is not bigger than 0.1.

```
a<c && c<b || b-a<=0.1
```

## Question 2: (20 points)

**Part (a):** (10 points)

For each fragment below, write a **`while`**-loop to produce the same printed output as the given **`for`**-loop. Only the output needs to be the same—any intermediate values need not be the same.

```
for k= 100:-2:5
   disp(k)
end
```

```
k= 100;
while k>=5     % or k>5, k>=6
    disp(k)
    k= k-2;
end
```

---

```
for k= 5:2:100
   disp(k)
   k= 200;
end
```

```
k= 5;
while k<=100   % or k<100, k<=99
    disp(k)
    k= k+2;
end
```

**Part (b):** (10 points)

Let ***x*** and ***n*** be variables that store positive integer values. Without using the ^ operator or the equivalent built-in function `power`, write a fragment to print the values $x$, $x^2$, $x^3$, …, $x^n$. Any print format is acceptable.

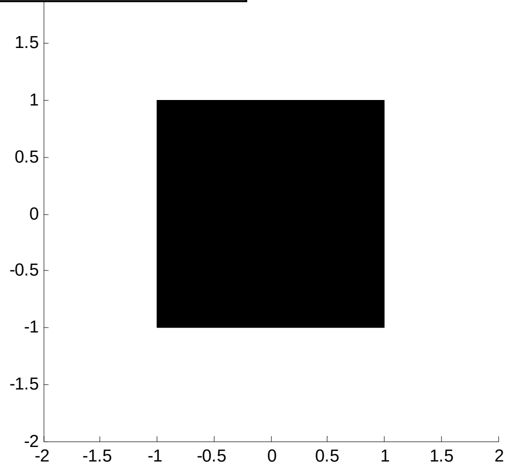
```
val= 1;
for k= 1:n
    val= val*x;
    disp(val)
end
```

Grading notes:
OK to print by leaving out semi-colon
Check that boundary case n=1 is correct

## Question 3:  (20 points)

*The question has been revised to draw another square instead of a disk.*

The script below produces the figure shown on the right. Add code to the script to add another square centered at a randomly generated position (x,y). x and y are uniformly random in the range of -1.5 to 1.5. The square has side length 0.8 (so it is inside the axis limits). The square should be red if it lies completely inside the black square, blue if it lies completely outside the black square, or magenta if it crosses any edge of the black square. You may assume that the smaller square is never tangent to any edge.



Assume the availability of function **`DrawRect`**:

```
DrawRect(-1,0,2,3,'m')
```

draws a magenta rectangle that has length 2 and height 3 with its lower left corner at (-1,0).

---

```
close all;  figure;  axis equal;  axis ([-2 2 -2 2]);  hold on

DrawRect(-1,-1,2,2,'k')
r= 0.8;  % SIDE LENGTH of SQUARE to be drawn

% (x,y) is the position of the center of the SQUARE to be drawn.
% Generate two uniformly random values in (-1.5,1.5) for x and y.


x= rand(1)*3-1.5;
y= rand(1)*3-1.5;


% Draw the SQUARE in the appropriate color

if abs(x)+r/2<1 && abs(y)+r/2<1      % inside black square

    DrawRect(x-r/2,y-r/2,r,r,'r')

elseif abs(x)-r/2>1 || abs(y)-r/2>1  % outside black square

    DrawRect(x-r/2,y-r/2,r,r,'b')

else                                 % on edge of black square

    DrawRect(x-r/2,y-r/2,r,r,'m')
end



hold off
```

Some students literally replaced the DrawDisk function with an equivalent DrawSquare function. The parameter "radius" is then the "half length" of the square, which is r/2 in the code. We accepted it.

```
% Draw the SQUARE in the appropriate color

if abs(x)+r/2<1 && abs(y)+r/2<1      % inside black square

    DrawSquare(x,y,r/2,'r')

elseif abs(x)-r/2>1 || abs(y)-r/2>1  % outside black square

    DrawSquare(x,y,r/2,'b')

else                                 % on edge of black square

    DrawSquare(x,y,r/2,'m')
end
```

## Question 4: (20 points)

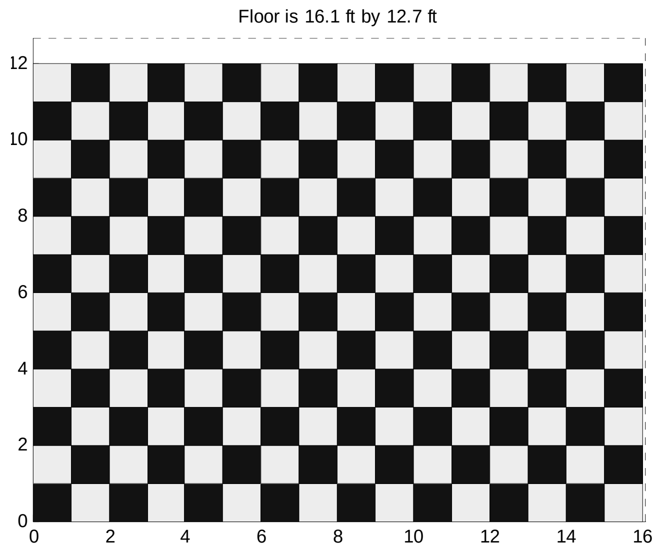
Complete the script below to simulate a game in which a person is given two sticks and must select a third from a random bag of sticks until he or she has three sticks that can form a triangle. The lengths of the first two sticks, **a** and **b**, are fixed throughout the game. The sticks in the "random bag" have lengths that are uniformly random in (0,1). The game ends when the player can form a triangle with three sticks or after the player has drawn 50 sticks from the bag, whichever happens first. Display the number of sticks the player has drawn from the random bag by the end of the game.

---

```
% The two given sticks have lengths a and b.  a and b do not change.
a= rand(1);
b= rand(1);

% Randomly generate c, the length of the third stick, in (0,1).  The
% simulation ends when it is possible to form a triangle with sticks a,b,c
% or after 50 attempts, whichever happens first.


maxTrials= 50;

c= rand(1);

k= 1;  % #attempts so far

while (a+b<=c || a+c<=b || b+c<=a) && k<maxTrials


    c= rand(1);

    k= k+1;
end

fprintf('Made %d attempts.\n', k)
```

Grading notes:
Check that printed k is correct if the first attempt works.
Check that there is never a 51st attempt.

## Question 5: (30 points)



Complete the script to draw the tiled floor pattern shown on the right using the **`DrawRect`** function. Each tile is one foot by one foot and only whole tiles are used. The bottom left corner tile is dark. The floor dimensions in the diagram on the right are only an example run of your script—the actual floor dimensions are input by the user. You may assume that the user enters positive values for the dimensions. If the floor is <1 ft$^2$ then no tile should be drawn.

Assume the availability of function **`DrawRect`**

```
          DrawRect(5,0,1,2,'b')
```

draws a blue rectangle that has width 1 and height 2 with its lower left corner at (5,0). You may use any dark and light color tile combination, e.g., blue ('b') and yellow ('y') tiles.

---

```
close all;  figure;  axis equal;  hold on

% User inputs floor dimensions
right= input('Enter a positive value: ');
top= input('Enter a positive value: ');

% Draw bounding box from (0,0) to (right,top) to represent floor
plot([0 right],[top top],'k:')
plot([right right],[0 top], 'k:')
title(sprintf('Floor is %.1f ft by %.1f ft', right, top))

% Draw the tiles in the given pattern.  Bottom left corner has a dark tile.


% Lower left corner of a tile is at (x,y)

for x= 0:1:right-1

    for y= 0:1:top-1

        if rem(x,2)==0 && rem(y,2)==0 || ...
           rem(x,2)==1 && rem(y,2)==1
           % Blue tile when both x and y are even or
           %           when both x and y are odd

            DrawRect(x,y,1,1,'b')
        else
            DrawRect(x,y,1,1,'y')
        end
    end
end


hold off
```

```
% Alternate strategy:
% Draw one big light color rectangle and then put dark tiles on top

DrawRect(0,0,floor(right),floor(top),’y’)

for x= 0:2:right-1

    for y= 0:2:top-1

        if rem(x,2)==0 && rem(y,2)==0 || ...
           rem(x,2)==1 && rem(y,2)==1
           % Blue tile when both x and y are even or
           %           when both x and y are odd

           DrawRect(x,y,1,1,'b')
        end
    end
end
```