Things to watch out for in upcoming exams (and homework). We WILL take points off for these in the future.

- Do not comment every line of code! Comments are for clarity; commenting every line just clutters things up! Using meaningful variable names typically reduces the commenting need. Observe the level of commenting shown in these solutions. Do not write meaningless comments such as "this is a loop" or "loop will stop when this condition isn't true"…

- Do not put a semi-colon after an if-CONDITION or an elseif CONDITION:

```
if  x>y ;
   blah
elseif  x>z ;
   blah blah
end
```

BAD!! No output echo is produced by Matlab here, so there's no need for the semi-colon. (Output echo is produced only for assignment operations.) And if you like ending a STATEMENT with a semi-colon, then do so only at the end of a statement. "if x>y" on its own is NOT a complete statement (the statement ends after "blah" in this example). This extra semi-colon is poor style in Matlab, but in another language it would be an actual error!

Question 1: (10 points)

Part (a): (3 points)

What are the final values of variables x and y?

```
x= 5;
y= 2*x;
x= 8;
```

x: 8

y: 10

Part (b): (3 points)

What are the final values of variables x and y?

```
x= 12;  y= 1;
if x>5
   x= x-y;
elseif x>10
   x= y;
else
   y= x;
end
```

x: 11

y: 1

Part (c): (4 points)

What is the output produced by this script?

```
x= 1;
for k= 8:-2:3
   x= x+k;
   disp(x)
end
```

9
15
19

Question 2: (15 points)

Part (a): (6 points)

Write a fragment to simulate one flip of an unfair coin: heads appears three times as often as tails. Assign to variable **c** the value 1 to indicate heads, the value 0 to indicate tails.

```
if  rand(1)<.75
    c= 1;
else
    c= 0;
end
```

Part (b): (9 points)

Write a fragment so that it behaves like the one below. *Do not* use any logical or bitwise operators (~, &&, ||, &, |). Assume that variables **a**, **b**, and **c** are initialized.

```
if a>b && b>c
   x= 2;
elseif b>a || a==8
   x= 4;
end
```

```
if  a>b
   if  b>c
      x= 2;
   elseif  a==8
      x= 4;
   end
elseif  b>a
   x= 4;
elseif  a==8
   x= 4;
end
```

```
if  b>a
   x= 4;
end
if  a==8
   x= 4;
end
if  a>b
   if  b>c
      x= 2;
   end
end
```

Tough question! We were lenient with this question. If you were very close to the solution (e.g., missing the inner elseif branch on the solution on the left), we gave you full credit.

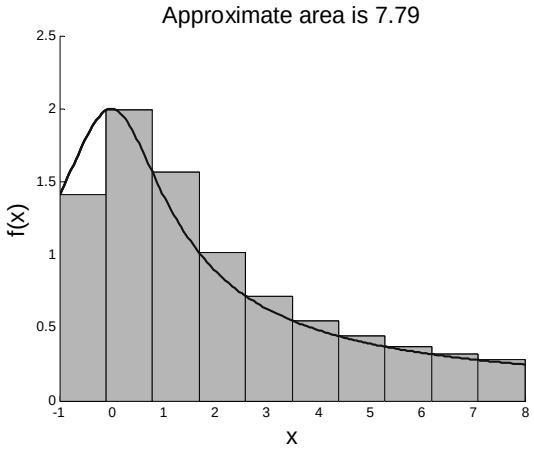
Question 3: (25 points)

Simulate a guessing game that involves a fair coin. A player starts by guessing how many flips of the coin will be made. Then the player flips the coin until he or she gets at least five heads or five tails, and the number of heads and the number of tails differ by no more than two. A message should be displayed at the end to indicate whether the player guessed correctly, e.g., "You are right!" or "Wrong!" Complete the script below to simulate one game.

```
% Coin game

guess= input('How many flips do you think you will need? ');

% Start flipping...



H= 0;  % number of heads so far
T= 0;  % number of tails so far

% GRADING NOTES:
% stop when        (H>=5 || T>=5) && abs(H-T)<=2
% continue when  ~((H>=5 || T>=5) && abs(H-T)<=2))
% continue when     H< 5 && T< 5  || abs(H-T)> 2

while  H<5 && T<5 || abs(H-T)>2

    if  rand(1)<.5

        H= H + 1;
    else

        T= T + 1;
    end
end

% Result
if  H+T == guess

    disp('You won!')
else
    disp('You lost!')
end
```

This question was mostly well done. The most common error was switching the && and || operators. Some students showed very cumbersome algorithm/structure, however. On this exam we didn't take points off for inefficiency or superfluous code (unless it was *really* bad), but we will progressively place more weight on such matters in the future!

Question 4: (25 points)



Complete the script below to approximate the area under the function $f(x) = \frac{2}{\sqrt{x^2+1}}$ for some interval of *x*. You will sum the area of *N* rectangles drawn side-by-side within the specified *x* interval with the *top left corner* of each rectangle at the function value. See diagram. Assume the availability of function **`drawRect`**:

`drawRect(-1,0,10,5,'g')`

draws a green rectangle with width 10, height 5, and the lower left corner at (-1,0). Your code should draw only the rectangles, not the curve. The curve is shown in the diagram only for clarity.

```
% Numerical integration--area of rectangles "under" a curve

L= input('Left end of the x interval: ');
R= input('Right end of the x interval: ');
N= input('Number of rectangles for approximation: ');

figure
axis([L R 0 2.5]); xlabel('x'); ylabel('f(x)')
hold on

% Code to draw the rectangles and approximate the area.  Remember to
% write on the blank near the bottom the variable name for the area.


area= 0;     % area of rectangles drawn so far

w= (R-L)/N;  % width of rectangles

for k= 1:N

    x= L+(k-1)*w;

    y= 2/sqrt(x^2+1);

    drawRect(x,0,w,y,'g')

    area= area + w*y;
end
```

Well done in general!

```
title(sprintf('Approximate area is %.2f', area ))
hold off
```

Question 5: (25 points)

Complete the script below to print in the *Command Window* a slanted U-figure (parallelogram without the top edge) formed by asterisks (*****) and blanks. Each side of the U-figure has **n** asterisks. You must use **`fprintf`** statements to print the figure to the *Command Window*. An example is shown below for **n** = 5. You may assume that **n** is greater than 2.

```
*   *
 *   *
  *   *
   *   *
    *****
```

```
% Print a slanted U
n= input('Enter an integer greater than 2:  ');



for r= 1:n
    for c= 1:r+n-1
        if  c==r || c==r+n-1 || (r==n && c>=n)
            fprintf('*')
        else
            fprintf(' ')
        end
    end
    fprintf('\n')
end

%%%%%% Alternate Solution %%%%%%%%%%%%%%%%%%%%%%%%

%All rows except last
for r= 1:n-1
    %leading spaces.  how many?  r-1
     for k= 1:r-1
        fprintf(' ')
     end
    %1st star
     fprintf('*')
    %n-2 middle spaces
     for k= 1:n-2
        fprintf(' ')
     end
    %last star
     fprintf('*\n')
end

%bottom row
for k= 1:n-1
    fprintf(' ')
end
for k= 1:n
    fprintf('*')
end
fprintf('\n')
```

Just like the section exercise/solution!