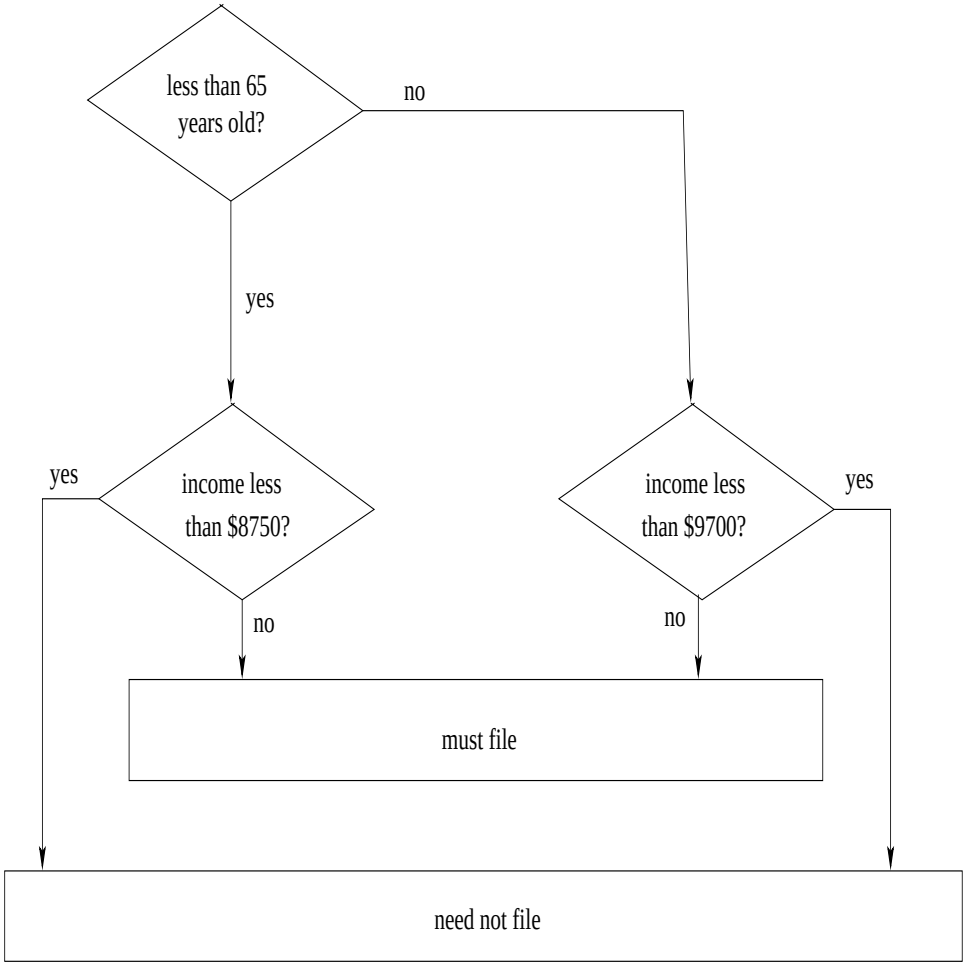
Question 2. *(4 pts.)*
Assume that the following variables describe a person and have already been assigned values:

- `age` is the person's current age in years
- `income` is the person's gross income

Write a MATLAB script that will display "`must file`" or "`need not file`" in the Command Window based on the value of these variables.

The logic that determines whether to file a tax return is given by this flowchart:



Solution:

```
if age < 65
    if income < 8750
        disp(’need not file’)
    else
        disp(’must file’)
    end
else
    if income < 9700
        disp(’need not file’)
    else
        disp(’must file’)
    end
end
```

Question 3. *(9 pts.)*
Coca Cola Corp. requires retailers to purchase the less popular Sprite product if the retailer wants to purchase the more popular Coke product.

Suppose that if the retailer orders 10 or fewer cases of Coke then it must buy 3 cases of Sprite. For such an order, each case of Sprite costs $12 while each case of Coke costs $18.

If the retailer orders more than 10 but fewer than 100 cases of Coke then it must buy 20% as many cases of Sprite, with the number of Sprite cases rounded up to the next integer. For such an order, each case of Sprite costs $11 while each case of Coke costs $15.

If the retailer orders 100 or more cases of Coke then it must buy 10% as many cases of Sprite, with the number of Sprite cases rounded up to the next integer. For such an order, each case of Sprite costs $10 while each case of Coke costs $12.

Write a MATLAB function named "`sodaCost`" that takes one argument and returns one output. The input is a positive integer number of Coke cases that a retailer orders. The output is the cost of all the Coke and Sprite cases that the retailer must purchase.

You can assume that the argument is good; you don't have to verify that it is a positive integer. Because of the limited available time, you need not document your function. MATLAB's `ceil` function rounds up to the nearest integer; e.g., `ceil(7.3)` returns 8.

Solution:

```
function [ cost ] = sodaCost( coke )

if coke <= 10
    cost = coke*18 + 3*12;
elseif coke < 100
    cost = coke*15 + ceil(coke*0.2)*11;
else
    cost = coke*12 + ceil(coke*0.1)*10;
end

end
```

Question 4. *(3 pts.)*
What are the values of `product`, `sum`, and `i` following execution of this MATLAB code?

```
product = 1;
sum = 0;
for i = 3:3:16
    if rem(i, 2) == 0
        product = product * i;
    else
        sum = sum + i;
    end
    i = i + 1;
end
```

`rem` is a MATLAB function that takes two integer arguments and returns the integer remainder generated when the first argument is divided by the second argument; for example, `rem(8, 3)` returns 2.

Solution: product = 72, sum = 27, i = 16.

`3:3:16` *creates the vector* `[ 3 6 9 12 15 ]`. *Variable* `i` *is set to these values at the top of each iteration. Therefore, the statement* `i = i + 1` *has no effect except at the end of the last iteration. During the last iteration,* `i` *is set to 15 at the top of the body then incremented to 16 at the bottom of the body.*

Question 5. *(5 pts.)*
Write a MATLAB script that plots the function $y = x^2 + 1$ at each whole number and half number in the interval $-8 \leq x \leq 8$. The X axis should be labeled "Input." The Y axis should be labeled "Output." The plot should be titled "A quadratic function."

Solution:

```
X = -8 : 0.5 : 8;
Y = X.^2 + 1;
plot(X, Y)
xlabel('Input')
ylabel('Output')
title('A quadratic function')
```

Question 6. *(4 pts.)*
Write a recursive MATLAB function named "`maxrec`" to compute the maximum element in a vector of numbers. The function must be recursive. Because of the limited available time, you need not document your function.

Solution:

```
function [ biggest ] = maxrec( v )

N = length(v);

% base case
if N == 1
    biggest = v(1);
    return
end

first = v(1);
rest = maxrec(v(2:N));
if first > rest
    biggest = first;
else
    biggest = rest;
end

end
```

There are other ways to define the recursive case, such as computing the maximum of the first N-1 numbers rather than the last N-1 numbers, or computing the maximum of the two halves, etc.

Question 7. *(7 pts.)*
In high school you studied the quadratic equation:

$$ax^2 + bx + c = 0$$

When a, b, and c are all zero the solution to this equation is "indeterminate" because any x value will satisfy the equation. When a and b are zero, there is no solution. When only a is zero, the solution is $-c/b$. In all other cases, the solutions are determined by the value of the *discriminant*, which is $b^2 - 4ac$:

- If $b^2 - 4ac < 0$ then the two solutions are complex
- If $b^2 - 4ac = 0$ then the single solution is $-b/2a$
- If $b^2 - 4ac > 0$ then the two solutions are real and distinct: $\frac{-b+\sqrt{b^2-4ac}}{2a}$ and $\frac{-b-\sqrt{b^2-4ac}}{2a}$

Write "pseudocode" that uses an intuitive mixture of programming language notation and English to describe the algorithm for computing and displaying the solutions to a quadratic equation, given the values a, b, and c. If no numerical solution(s) can be computed, your pseudocode should indicate that fact.

You do not need to write complete MATLAB code. However, your pseudocode should clearly address all the issues that would arise when writing a MATLAB program so that any skilled MATLAB programmer could easily write the corresponding MATLAB code based on your description.

Solution:
if a = b = c = 0, indicate error – indeterminate
if a = b = 0, indicate error – no solution
if a = 0, return -c/b
*compute discrimnant b*b - 4ac*
if discriminant < 0, indicate solutions are complex
if discriminant = 0, return -b/2a
if discriminant > 0, return two real solutions (-b +/- sqrt(discriminant)) / 2a

Question 8. *(5 pts.)*
Write a MATLAB function named "`diagonal`" that takes one argument and returns one output. The argument is a 2-dimensional square matrix of numbers. (A square matrix has equal sizes in each dimension, such as 2x2, 3x3, 4x4, etc.) The output is a vector that contains all the elements from the matrix's diagonal. For instance, if the argument matrix is 3x3 then the output vector will have three elements that are elements `(1, 1)`, `(2, 2)`, and `(3, 3)` from the matrix. As another example, if the argument matrix is $\begin{bmatrix} 1 & 5 \\ 7 & 3 \end{bmatrix}$ then the output vector will be `[ 1 3 ]`. Because of the limited available time, you need not document your function.

Solution:

```
function [ vec ] = diagonal( matrix )

N = length(matrix);

for i = 1:N
    vec(i) = matrix(i,i);
end

end
```