Name: ____ NetID: __________

(Legibly print last name, first name, middle name)

Statement of integrity: *It is a violation of the Code of Academic Integrity to look at any exam other than your own, to look at any other reference material, or to otherwise give or receive unauthorized help. Understanding this, I declare I shall not give, use, or receive unauthorized aid in this examination.*

____

(Signature)

Circle your lecture time: 9:05 or 11:15

Circle your discussion instructor's name:

	Tuesday	Wednesday
10:10		Noam Eshed
11:15		Noam Eshed
12:20	Zhilong Li	Joseph Kim
1:25	Vaishnavi Dhulkhed	Joseph Kim
2:30	Darian Nwankwo	Darian Nwankwo
3:35	Darian Nwankwo	Zhilong Li

Instructions:

- This is a 90-minute, closed-book exam; no calculators are allowed.
- The exam is worth a total of 100 points, so it's about one point per minute!
- Read each problem completely, including any provided code, before starting it.
- Raise your hand if you have any questions.
- Clarity, conciseness, and good programming style count for credit.
- If you supply multiple answers, we will grade only one.
- Use only MATLAB code. No credit for code written in other programming languages.
- Assume there will be no input errors.
- Do not modify given code unless instructed to do so.
- Do not write user-defined functions or subfunctions unless instructed to do so.
- Do not use `switch`, `try`, `catch`, `break`, `continue`, or `return` statements.
- Do not use built-in functions that have not been discussed in the course.
- You may find the following MATLAB predefined functions useful:
  `abs`, `sqrt`, `rem`, `floor`, `ceil`, `round`, `rand`, `zeros`, `ones`, `linspace`, `length`, `input`, `fprintf`, `disp`, `bar`

Examples:

`rem(5,2)` → 1, the remainder of 5 divided by 2

`rand(1,3)` → 1 row 3 columns of numbers, each uniformly random in the open interval (0,1)

`floor(6.9)`, `floor(6)` → 6, rounds down to the nearest integer

`ceil(8.1)`, `ceil(9)` → 9, rounds up to the nearest integer

`zeros(1,4)` → 1 row 4 columns of zeros

`length([2 4 8])` → 3, length of a vector

Question 1: (17 points)

(a) Write in the box below the first **six** lines of output that would be produced by executing the following script. Write on the blank the total number of lines of output that would be produced by the script.

```
for t = 1:4
    for u = 5:6
        disp(u-t)
        v(u-t)= t+10;
    end
    disp(v)
end
```

First 6 lines of output:

```
4
5
0 0 0 11 11
3
4
0 0 12 12 11
```

How many lines of output will be produced in total? **12**

(b) Suppose you are charged one dollar for every relational operation that *is executed.* (Note: `>=` and `~=` are examples of relational operators. `&&` and `||` are *not* relational operators; they are logical operators.) What is the output and how much is the fee that you have to pay after executing the following script?

```
a=9; b=7; c=9;
if (a>=c && c>b) || a==0
    disp('D')
elseif c>b
    disp('E')
end
if b>=c || a==0
    disp('F')
end
```

Output: **D**

Fee (in dollars): **4**

(c) What will be printed when the following script is executed? Use the specified print format.

Script	*Function*
`g= [2 4];`	`function h = fun(g,r)`
`h= 10;`	`g(1)= g(2);`
`s= fun(g,h);`	`g(2)= g(1);`
`fprintf('s is %d\n', s)`	`h= r-g(1);`
`fprintf('g(1) is %d\n', g(1))`	`fprintf('g(2) is %d\n', g(2))`
`fprintf('h is %d\n', h)`	`fprintf('h is %d\n', h)`

Output:

```
g(2) is 4
h is 6
s is 6
g(1) is 2
h is 10
```

Question 2: (15 points)

(a) Complete the loop condition below so that the user is prompted until they enter an odd integer between 1 and 35 (inclusive).

```
  n = input('Place an odd bet (1-35): ');

  while __ (n < 1) || (n > 35) || (rem(n, 2) ~= 1) ___
     n = input('Place an odd bet (1-35): ');
  end
```

(b) Write a function named `recomputeEvens` that has one input parameter `v` (a vector) and returns a vector of the same length whose odd-indexed elements match those of `v` but whose even-indexed elements are equal to the average of that element's left and right neighbors. You may assume that `v` has an odd length of at least 3.

Example: `recomputeEvens([1 -2 4])` should return `[1 2.5 4]`.

Note: You must write the function header along with the function body, but you do not need to write the function comment.

```
function v = recomputeEvens(v)
for k = 2:2:length(v)
    v(k) = (v(k - 1) + v(k + 1))/2;
end


%% Alternate solution
%function w = recomputeEvens(v)
%for k = 1:length(v)
%    if rem(k, 2) == 1
%        w(k) = v(k);
%    else
%        w(k) = (v(k - 1) + v(k + 1))/2;
%    end
%end
```

Question 3: (25 points)

(a) Assume income tax is levied as follows: the first \$20,000 of a person's income is not taxed, then the next \$30,000 is taxed at 25%, the next \$50,000 above that is taxed at 30%, and finally any income over \$100,000 is taxed at 40%. As an example, a person earning \$60,000 of income would owe \$10,500 in tax ($0 \times \$20\,000 + 0.25 \times \$30\,000 + 0.3 \times \$10\,000$), for an *effective rate* (tax divided by income) of 17.5%, i.e., 0.175. Implement the following function to compute someone's tax burden (the built-in functions `rem`, `min`, and `max` are allowed if you want to use them, but they are not necessary):

```
function [t, r] = incomeTax(income)
% Compute the amount of tax, t, owed on income dollars, and also compute the
% effective tax rate, r, a value between 0 and 1.  Do not worry about rounding
% money to the nearest cent.
```

```
tax = 0;
remaining = income - 20000;
if remaining > 0
    tax = tax + 0.25*min(remaining, 30000);
    remaining = remaining - 30000;
end
if remaining > 0
    tax = tax + 0.3*min(remaining, 50000);
    remaining = remaining - 50000;
end
if remaining > 0
    tax = tax + 0.4*remaining;
end
rate = tax/income;


%% Alternate approach, with misinterpretation of highest bracket
%if income <= 20000
%    tax = 0;
%elseif (income - 20000) <= 30000
%    tax = 0.25*(income - 20000);
%elseif (income - 20000 - 30000) <= 50000
%    tax = 0.3*(income - 20000 - 30000) + 0.25*30000;
%else
%    tax = 0.4*income;
%end
%rate = tax/income;
```

(b) There is a proposal to introduce a new tax bracket affecting incomes over $250,000. Write a script to analyze the impact of this change. Assume a vector `incomes` is present in the Workspace containing the incomes (in dollars) of everyone in the district. Your script should print the number of people who would be affected by this change, as well as the highest effective rate currently levied on one of them. You should call function `incomeTax` (and may assume it has been implemented correctly) from part (a), but the only built-in function you may use is `length`.

```
% Assume vector incomes is given; it is not empty.
% Write your code below.
```

```
highestRate = 0;
affectedCount = 0;
for income = incomes
    if income > 250000
        affectedCount = affectedCount + 1;
        [tax, rate] = incomeTax(income);
        if rate > highestRate
            highestRate = rate;
        end
    end
end
fprintf('Affected persons: %d\n', affectedCount)
fprintf('Highest rate: %.3f\n', highestRate)
```

Question 4: (18 points)

(a) Implement the following function as specified:

```
function rv = reverseVector(v)
% rv has the same elements as vector v but with the order reversed.
% Do NOT use vectorized code.  The only built-in function allowed is length.
```

Example solution 1:

```
n= length(v);
rv= zeros(1, n);  % init not necessary
for k= 1:n
    rv(n-k+1)= v(k);
end
```

Example solution 2:

```
n= length(v);
j= n;
for k= 1:n
    rv(j)= v(k);
    j= j-1;
end
```

Example solution 3:

```
n= length(v);
for k= 1:n/2
    % swap v(k) with v(n-k+1)
    temp= v(k);
    v(k)= v(n-k+1);
    v(n-k+1)= temp;
end
rv= v;
```

(b) The binary representation of a number is expressed in the base-2 numeral system using only 0s and 1s. For example, the binary representation of 5 is [1 0 1] because $5 = \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0$. The 0s and 1s are ordered left to right starting with "the most significant bit," i.e., the coefficient (0 or 1) of the highest power of 2 needed, and ending with the least significant bit, i.e., the coefficient of the 0th power of 2. To compute the binary representation (vector) of a positive integer number, you will work backwards, finding the least significant bit first:

- Divide the number by 2. Append the remainder to the results vector, and replace the number with the integer quotient.
- Repeat the above step as long as the number is positive.
- Reverse the order of the elements in the vector.

As an example, below is the process for computing the binary representation of the number 12:

Division	*Quotient*	*Remainder*	*Vector*
12/2	6	**0**	[**0**]
6/2	3	**0**	[0 **0**]
3/2	1	**1**	[0 0 **1**]
1/2	0	**1**	[0 0 1 **1**]

Reverse the vector to get the binary representation: [1 1 0 0]

By making effective use of function `reverseVector` from Part (a), implement the following function using the algorithm given above:

```
function v= binaryVector(num)
% Vector v is the binary representation of positive integer num.
% The only built-in functions allowed are rem, floor
```

```
k= 1;  % v= zeros(1,floor(log2(num))+1)
while num>0
    v(k)= rem(num,2);
    k= k+1;
    num= floor(num/2);
end
v= reverseVector(v);
```

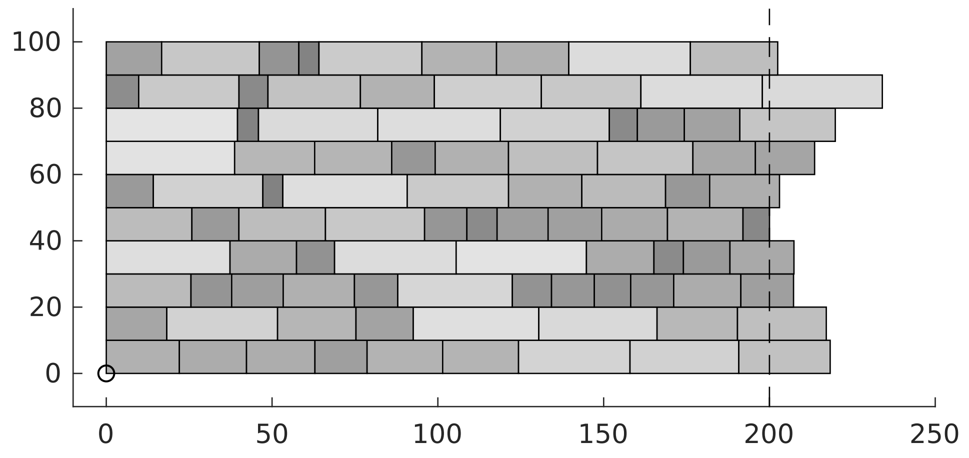
Question 5: (25 points)

Write a script that assembles and draws a stone wall using random bricks from a quarry. The end result should resemble the figure below, where stones have been colored according to their size. The parameters of the job are as follows:

- Stones are all 10 cm tall, but their widths are random, varying uniformly between 5 cm and 40 cm (*not* restricted to an integer number of cm)
- The wall must be at least 200 cm wide; stones may protrude past that edge, but no new stones should be placed after it
- The wall must be 100 cm (10 stones) high
- The skinniest (5 cm) stones should be colored dark grey, while the widest (40 cm) stones should be colored light grey; stones of intermediate width should have a color that linearly interpolates between these two shades
- The first stone should have its lower-left corner at coordinate $(0, 0)$

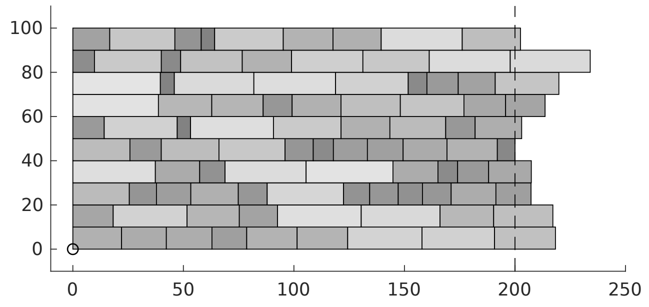
Assume the availability of a function `DrawRect(xLeft, yBottom, width, height, color)` that draws a rectangle whose lower-left corner is (`xLeft`, `yBottom`), whose width and height are `width` and `height`, and which is filled with the color specified by the 3-vector `color`. Your code only needs to draw the stones (axes and markers are shown above just for reference).

Hint: DECOMPOSE! First work on drawing a single row of white rectangles, then build up multiple rows, and finally revise your code to deal with different shades of grey.



Write your code on the next page.

[Question 5, continued]



```
% Script to assemble and draw a stone wall
figure; axis equal off; hold on
darkGrey = [0.5 0.5 0.5];
lightGrey = [0.9 0.9 0.9];
```

```
stoneHeight = 10;
wallWidth = 200;
numRows = 10;
yBottom = 0;
for row = 1:numRows
    xLeft = 0;
    while xLeft < wallWidth
        scale = rand();
        stoneWidth = (40 - 5)*scale + 5;
        stoneColor = darkGrey*(1 - scale) + lightGrey*scale;
        DrawRect(xLeft, yBottom, stoneWidth, stoneHeight, stoneColor);
        xLeft = xLeft + stoneWidth;
    end
    yBottom = yBottom + 10;
end
```

```
hold off
```