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 reference material outside of this exam packet, to communicate with anyone other than the exam proctors, or to otherwise give or receive any unauthorized help during the exam.

Academic Integrity is expected of all students of Cornell University at all times. **By submitting this exam, you declare that you will not give, use, or receive unauthorized aid in this examination.**

Circle your discussion section:

Section	Day and Time	Instructor
201	W 10:10 AM - 11:00 AM	Carlos Alvarez
202	W 11:20 AM - 12:10 PM	Carlos Alvarez
203	W 12:25 PM - 1:15 PM	Dominic Diaz
204	W 1:30 PM - 2:20 PM	Dominic Diaz
205	W 2:40 PM - 3:30 PM	Xinran Zhu
206	W 3:45 PM - 4:35 PM	Xinran Zhu

Instructions:

- Check that this packet has 7 double-sided sheets.
- 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.
- Do not modify any *given* code unless asked to do so.
- Raise your hand if you have any questions.
- Use the back of the pages if you need additional space.
- Clarity, conciseness, and good programming style count for credit.
- Indicate your final answer. If you supply multiple answers, you may receive a *zero* on that question.
- Use only MATLAB code. No credit for code written in other programming languages.
- Assume there will be no input errors.
- Write user-defined functions and subfunctions only if asked 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`, `rand`, `zeros`, `ones`, `linspace`, `length`, `input`, `fprintf`, `disp`

Examples:
`rem(5,2)` → 1, the remainder of 5 divided by 2
`rand()` → a random real value in the interval (0,1)
`abs(-3)` → 3, absolute value
`floor(6.9)`, `floor(6)` → 6, rounds down to the nearest integer
`ceil(8.1)`, `ceil(9)` → 9, rounds up to the nearest integer
`length([2 4 8])` → 3, length of a vector
`zeros(1,4)` → 1 row 4 columns of zeros
`linspace(3,5,10)` → a vector of 10 real numbers evenly distributed in the interval [3,5]

Question 1 (16 points)

(1.1) What is the value of z after executing the following script?

```
z= 0;
for i= 1:10
    for j= 1:50
        if i==j
            z= z + 1;
        end
    end
end
```

Solution: 10

(1.2) What will be printed when the following script is executed?

Script	*Function* (in `foo.m`)
`a = 9;`	`function h = foo(a, b)`
`b = 5;`	`if a < 10`
`h = 1;`	`    h = a;`
`c = foo(b, a);`	`    a = 10;`
`fprintf('c is %d\n', c)`	`    fprintf('h is %d\n', h)`
`fprintf('a is %d\n', a)`	`end`
`fprintf('h is %d\n', h)`	`if 10 > b`
	`    h = b - 2;`
	`    fprintf('h is %d\n', h)`
	`end`

Solution:

```
h is 5
h is 7
c is 7
a is 9
h is 1
```

(1.3) Rewrite the following code fragment without using the logical operators `&&`, `||`, and `~` such that the variable `x` is assigned the same value as the given fragment. (Do not use `&`, `|`, or `xor` either.)

```
% Assume that variables a, b, c, d, e
% are each a numeric scalar

x= 0;
if  a==b && a>c
    x= 1;
elseif  b==d || b==e
    x= 2;
end

% Write your version in the box on
% the right
```

Example solution

```
x= 0;
if  a==b
    if a>c
        x= 1;
    elseif b==d  % No penalty if this branch is missed
        x= 2;
    elseif b==e  % No penalty if this branch is missed
        x= 2;
    end
else
    if b==d
        x= 2;
    elseif b==e
        x= 2;
    end
end
```

Question 2 (14 points)

(2.1) Add code to the script below to calculate the sum of the user input values and display the sum. You may add code above, within, and/or below the loop. Do not cross out any given code. Do not call any built-in functions other than `disp` or `fprintf`.

```
s= 0;  %%%% Solution
for k= 1:10
    num= input('Enter a number: ');
    s= s + num;  %%%% Solution
end
disp(s)  %%%% Solution
```

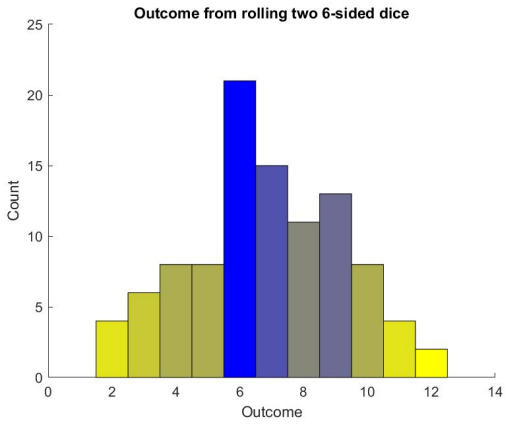
(2.2) Add code in the box below to compute the color of each bar of a histogram. *The provided code does everything else needed to draw the histogram,* with the data vector `h` assumed to be given as described.

Two fair, 6-sided dice were rolled 500 times and the outcomes were tallied in a length 11 vector `h` such that

- `h(1)` stores the number of times that the outcome 2 occurred,
- `h(2)` stores the number of times that the outcome 3 occurred, ..., and
- `h(11)` stores the number of times that the outcome 12 occurred.

The code fragment below is to draw a histogram of the outcomes. The tallest bar is blue, the shortest bar is yellow, and each bar has a color that is linearly interpolated between blue and yellow according to its height. An example histogram that could be produced is shown at the bottom of the page.

```
% Assume that h, the vector storing the count of the outcomes, is given
hold on
maxh= max(h);  % height of the tallest bar
minh= min(h);  % height of the shortest bar
blue=   [0 0 1];  % color of tallest bar
yellow= [1 1 0];  % color of shortest bar
for k= 1:11
    % The height of the kth rectangle is h(k).
    % Add code below to compute vector c, the color of the kth rectangle.

    %%%% Example solution
    frac= (h(k)-minh)/(maxh-minh);
    c= frac*blue + (1-frac)*yellow;
    %%%%

    % Draw the kth rectangle
    DrawRect(k+.5, 0, 1, h(k), c)
end
title('Outcome from rolling two 6-sided dice')
xlabel('Outcome'); ylabel('Count')
hold off
```



Question 3 (15 points)

To clamp a vector v to an interval L to U is to change any values in v greater than U to U and any values in v less than L to L. The other values in v remain unchanged.

Implement the following function as specified. For full credit, use a loop to solve this problem. Do not use vectorized arithmetic or vectorized comparison.

```
function [w, b] = clamp(v, L, U)
% Clamp vector v to the interval L to U.
% Input parameters:
%   v:   a non-empty vector to be clamped
%   L,U: each is a numeric scalar representing the interval for clamping, L<U
% Return parameters:
%   w:   the vector after clamping (modified from v)
%   b:   a vector, possibly empty, storing the values in v less than L or
%        greater than U
% Example:  clamp([ -2 3.1 0.5], 0.1, 3)  should return these two vectors:
%               w=[0.1   3 0.5]  and  b=[-2 3.1]

b= [];
for k= 1:length(v)
    if v(k) < L
        w(k)= L;
        b= [b v(k)];
    elseif v(k) > U
        w(k)= U;
        b= [b v(k)];
    else
        w(k)= v(k);
    end
end
```

Question 4 (15 points)

A non-empty vector is a zigzag vector if the values at odd indices are positive and the values at even indices are negative. Here are some examples:

- `[.2 -1 5]` is a zigzag
- `[.2 -1 0 -2]` is not a zigzag (0 is neither positive nor negative)
- `[-1 .2 -5]` is not a zigzag

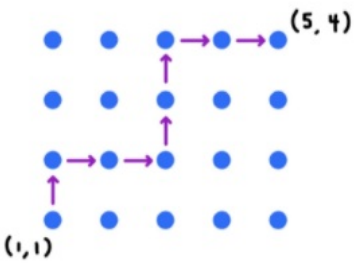
Implement the following function as specified. For full credit, make effective use of a while-loop in order to avoid unnecessary iterations. Vectorized code is forbidden.

```
function isZZ = isZigZag(v)
% Returns true (1) if vector v is a zigzag vector; otherwise false (0).
% v: a numeric vector with at least 3 elements
% isZZ: a boolean value, true or false (1 or 0)

isZZ= true;  % 1 also works
k= 1;
n= length(v);
while  k<=n  &&  isZZ
    if rem(k,2)==1 && v(k)<=0  %odd index but not positive
        isZZ= false;
    elseif rem(k,2)==0 && v(k)>=0  % even index but not negative
        isZZ= false;
    end
    k= k + 1;
end
```

Question 5

5.1 (20 points) Implement the following function as specified. For full credit, make effective use of a `while`-loop to compute the path. Note that an example is given at the bottom of the page.

```
function [xvec, yvec]= randPath(xlo, ylo, xhi, yhi)
% Returns a semi-random path from point (xlo,ylo) to point (xhi,yhi).
% xlo, ylo: each is an integer
% xhi, yhi: each is an integer such that xhi>xlo and yhi>ylo
% Compute each step of the path as follows:
%   Choose to go right or go up with equal likelihood, with these exceptions:
%   (1) If the current y-coordinate is yhi, then go right.
%   (2) If the current x-coordinate is xhi, then go up.
% xvec, yvec are vectors that together store the path, including the
%   starting and destination coordinates.  I.e., xvec(1)=xlo, yvec(1)=ylo,
%   the last element of xvec is xhi, and the last element of yvec is yhi.
%   Therefore the length of each vector is one plus the number of steps
%   taken to reach (xhi,yhi).
% For full credit, make effective use of a while-loop to compute the path.

xvec= xlo;
yvec= ylo;
k= 1;
while  xvec(k) < xhi || yvec(k) < yhi
    deltaX= 0;
    deltaY= 0;
    if xvec(k)==xhi
        deltaY= 1;
    elseif yvec(k)==yhi
        deltaX= 1;
    else
        if rand()<.5
            deltaY= 1;
        else
            deltaX= 1;
        end
    end
    k= k+1;
    xvec(k)= xvec(k-1) + deltaX;
    yvec(k)= yvec(k-1) + deltaY;
end
```



Example path from (1,1) to (5,4)
xvec = [1, 1, 2, 3, 3, 3, 4, 5]
yvec = [1, 2, 2, 2, 3, 4, 4, 4]

5.2 (20 points) For this part of the question, assume that function `randPath` from (5.1) is correctly implemented and is accessible. Furthermore, assume the availability of a function `numTurns` that returns the number of turns that a path takes given the x-coordinates and y-coordinates of the path as arguments. Here is an example call to `numTurns` using the path data of the example shown in (5.1):

```
x= [1 1 2 3 3 3 4 5];
y= [1 2 2 2 3 4 4 4];
nt= numTurns(x, y)  % nt is 3 since the path takes 3 turns (see diagram of (5.1))
```

Do not implement `numTurns`; just call it for the question below. Built-in functions `min`, `max`, and `sort` are forbidden.

Suppose that XX is a vector of 10 positive integers representing 10 destination x-coordinates and YY is a vector of 15 positive integers representing 15 destination y-coordinates. Then there are `length(XX)` $\times$ `length(YY)` $= 150$ possible destinations given XX and YY. Complete the code fragment below to call `randPath` to compute the path from point (0,0) to each of the 150 possible destinations given by XX and YY. Determine which path took the most number of turns. Assume that only one path had the maximum number of turns.

```
XX= [5 6 7 8 9 15 30 2 3 4];

YY= floor(rand(1,15)*(99-7+1)) + 7;
%    ceil(rand(1,15)* 93     ) + 6

maxTurns= 0;  % Can initialize as  -inf, any value <=0.
              % If initialized to the #turns of the first combo then may
              % need to initialize x_maxTurn and y_maxTurn as well.

for k= 1:length(XX)
    x= XX(k);
    for j= 1:length(YY)
        y= YY(j);
        [xvec,yvec]= randPath(0,0, x, y);
        nTurns= numTurns(xvec, yvec);
        if nTurns > maxTurns
            maxTurns= nTurns;
            x_maxTurn= x;
            y_maxTurn= y;
        end
    end
end

fprintf('The path to the destination (%d,%d) has the most turns.\n', ...
        x_maxTurn, y_maxTurn)
```