CS100M **September 25, 2000**
Prelim 1 **7:30 PM – 9:00 PM**

_____ _____
(Print last name, first name, middle initial/name) (Student ID)

Statement of integrity: I did not, and will not, break the rules of academic integrity on this exam:

_____
(Signature)

Circle Your Section:

	Tuesday			Wednesday	
	PH 403	PH 407	UH 111	HO 306	UH 111
1:25	1 Nagarajan		2 Fan	6 Rohde	
2:30		3 Nagarajan	5 Fan		7 Fernandes
3:35		4 Fernandes		8 Rohde	

Instructions:

- Read all instructions *carefully*, and read each problem *completely* before starting it!
- This test is closed book – no calculators, reference sheets, or any other material allowed.
- Initial or sign each page.
- Conciseness, clarity, and style all count. Show all work to receive partial credit.
- Carefully comment each loop and major variable.
- If *you* use **`break`** to exit any control structure, you will lose points!
- You may *not* alter the structures surrounding blanks and boxes.
- Only *one* statement, expression, or comment per blank!
- Use the backs of pages if you need more space or scrap. You may request additional sheets from a proctor.
- If you supply multiple answers, we will grade only *one*.

Core Points:

1. _____ (13 points) _____

2. _____ (13 points) _____

3. _____ (28 points) _____

4a. _____ (16 points) _____

4b. _____ (30 points) _____

Total:_____ (100 points) _____

Bonus Points:

_____/ (7 bonus points) _____

MATLAB Reminders

[] (square brackets), e.g,

```
>> v = [7  12  5]
v =
      7  12  5
```

() (parentheses), e.g,

```
>> v([1 2])
ans =
      7  12
```

: (colon), e.g,

```
>> 1:3
ans =
      1  2  3
```

relations (**<**, **>**, **<=**, **>=**, **==**, **~=**), boolean values (**0**, **1**), and logical operators (**|**, **&**), e.g,

```
>> 1 | 0
ans =
      1
```

abs (absolute value), e.g.,

```
>> abs([1 -2])
ans =
      1  2
```

max (maximum value), e.g.,

```
>> max(1:2)
ans =
      2
```

Problem 1 [13 points] *scalars, arrays, operators*

Fill in the boxes with the output that MATLAB will generate for the following statements.

```
% Problem 1.1 (3 points)
>> a = [1 2, 3; 4:0.5:5; [3;3;3]']
a =
```

```
% Problem 1.2 (2 points)
>> b(1,2) = 10
b =
```

```
% Problem 1.3 (4 points)
>> c = [1 2 1];
>> c(1, [2 1 1]);
>> c(1,c(1,[3 2])) = [2 1]
c =
```

```
% Problem 1.4 (4 points)
>> d = [1 2 3; 4 5 6];
>> d([2 1],:) =  [ 4:-1:2 ; d(5) d(2,[2 3])]
d =
```

Problem 2 [13 points] *logic, conditionals*

Fill in the blanks with the output that MATLAB will generate for the following statements.

```
% Problem 2.1 (2 points)
>> a = 1 == 2
a =
```

```
% Problem 2.2 (4 points)
>> u = 0; v = 0; w = 2;
>> b = (u >= ~w & ~v == w-1 & ~u >= w)
b =
```

```
% Problem 2.3 (3 points)
>> c = -~~3
c =
```

```
% Problem 2.4 (4 points)
>> d = -1;
>> if d
      disp(['d= ',num2str(d)])
   end
>> d = d+1;
>> if d
      disp(['d= ',num2str(d)])
   end
% Output:
```

Problem 3 [28 points] *I/O, conditionals*

A shipping company calculates the shipping prices as follows:

- A package weighing 5 lb (pound) or less costs $12, excluding tax.
- A package weighing over 5 lb and less than 10 lb costs $18, excluding tax.
- A package weighing at least 10 lb costs $20 plus $1.50 for each pound over 10 lb. For example, a 10.5 lb package costs $20.75 to ship, excluding tax.
- Tax (8%) is charged for shipment to Region 1. No tax is charged for shipment to Region 2.

Fill in the blanks and boxes below to complete a program that

- prompts the user for the package weight (**`weight`**) and destination code (**`code`**). The code must be entered as **`1`** or **`2`**.
- calculates the shipping charge (including tax) and stores the result in **`charge`**.

You do not need to check for non-numerical inputs, like strings. You do not need to output **`charge`**, **`code`**, or **`weight`**.

```
% Obtain package $weight$
  weight = input('Enter weight: '); % Assume user enters a _____________________ value

% Obtain destination $code$
  code = input('Enter code [1 or 2]: ');

  while ____________________________________________

     disp(['What?'])
  code = input('Re-enter code: ');
  end

% Determine $charge$ based on $weight$
```

```
% Modify $charge$ based on destination $code$
```

Problem 4 [46 points] *arrays, loops, conditional update*

The following two problems (4a and 4b) solve for the maximum *gradient* from two elevation values. The gradient *g* between neighboring elevations *e1*and *e2* is $g = |e1 - e2|$, assuming constant spacing between measurements. Both problems solve a similar problem, but follow different approaches and perform different tasks, as explained in the problem descriptions.

4a) [16 points] Fill in the blanks below to complete the code for finding the maximum gradient from a vector of positive elevations. Follow the instructions inside the comments. Hints: you will need the **max** and **abs** functions from page 1 of this exam. To help write the code, try the algorithm on a simple vector, like [1 2 4]. Assume strictly numerical input.

```
% Array-oriented approach
% ----------------------
% Gather elevations in vector $ev$
  n = 0;                                % count of elevations so far
  e = input('Enter an elevation: ');    % enter first elevation
  while(e >= 0)                         % user must enter only positive elevations
     n = n+1;                           % increment count
     ev(n) = e;                         % store elevations in $ev$
     e = input('Enter an elevation: '); % process next input
  end

% Stop execution if no elevations entered
  if n==0
     disp(['Instituting instructor's privilege to use "evil" break.')
     break
  end

% Subtract the elevation vector from itself such that the resulting vector $g$:
% + stores the entire vector of gradients
% + contains one less element than L
% Remember to assume constant spacing between measurements.
% Use the $abs$ function to ensure all gradients in $g$ are positive
  g = _____ (____________________ - ____________________)

% Report maximum value inside $g$
  _____( ______ )

% Plot elevations (no titles or axis labels are required)
  plot(___________)
```

[This space intentionally left blank. Problem 4b is on the next page.]

4b) [30 points] Fill in the blanks below to complete the code for finding the maximum ("steepest") gradient and the two elevations that produce it. Follow the instructions inside the comments. You do not need to plot elevations in this problem, but you do need to output the maximum gradient and the elevations that produce it. Assume strictly numerical input.

```
% Conditional-Update approach
% --------------------------
maxGrad = 0;                            % steepest gradient found so far
elevL = 0;                              % left neighbor of steepest gradient so far
elevR = 0;                              % right neighbor of steepest gradient so far
current  = input('Enter elevation: '); % current elevation value
previous = current;                     % previous elevation value
gradient = abs(previous-current);       % gradient between current and previous elevs

% Process all elevations and find the following:
% + maximum gradient
% + the neighboring elevations that produce the maximum gradient
  while _____________________ % stop loop when user enters a negative value
     % If computed gradient is greater than maxGrad,
     % update $maxGrad$, $elevL$, $elevR$
       gradient = abs(_____________________ - _____________________);
       if _____________________ > _____________________
          maxGrad = _____________________________________;
          elevL = _____________________________________;
          elevR = _____________________________________;
       end
     % Update previous and read next input (current)
       _____________________ = _____________________ ;
       _____________________ = input('Enter elevation: ');
   end
% Output results: maximum gradient and the elevations that produce it
  if _____________________ == _____________________
     disp(['Only one elevation entered or all elevations are equal'])
  else
     disp(['The steepest gradient of ' num2str(_____________________)...
           'occurs between elevations '...
           num2str(__________) ' and ' num2str(__________) ])
  end
```

Checklist: Congratulations! You reached the last page of Prelim 1. Make sure your name, ID, and section are CLEARLY indicated. Also, re-read all problem descriptions/code comments/instructions. If you reached this part before exhausting the allotted time, check your test! Have you done the following?

- Completed all tasks
- Filled in ALL required blanks
- Given comments when necessary
- Declared all variables
- Maintained case-sensitivity
- Handled "special cases" correctly
- Used correct array bounds
- Indicated which solution to grade if you wrote multiple attempts

---

Bonus: [9 bonus points, 1 point each blank] (Remember that bonus points do not count towards your core-point total!)

Your lecture professor's middle initial **I** is short for _____.

MATLAB stands for _____ _____.

According to your lecture professor, programming is _____ _____ _____.

AEW stands for _____ _____ _____.

[This space intentionally left blank.]