CS100M Prelim 2

October 17, 2000
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.
- 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. ________ (12 points) _________
    2. ________ (42 points) _________
   3a. ________ (21 points) _________
   3b. ________ ( 4 points) _________
   3c. ________ (14 points) _________
   3d. ________ ( 7 points) _________
Total:________ (100 points) _________
```

Bonus Points:

```
      ______/ (9 bonus points) _______
```

MATLAB Reminders

`[]` (square brackets), e.g,

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

`[]` (empty array), e.g.,

```
>> v = []
v =
      []
```

`()` (parentheses), e.g,

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

`{}` (braces), e.g.,

```
>> w(2) = {'hi'}
w =
      []    'hi'
```

`.` (dot), e.g.,

```
>> z.x = 1
z =
      x: 1
```

`:` (colon), e.g,

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

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

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

`abs` (absolute value), e.g.,

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

`any` (return **`1`** if any element in a vector is non-zero), e.g.,

```
>> any([0,0,1,0])
ans =
      1
```

`char` (character), e.g.,

```
>> char(107)
ans =
k
```

`end` (for ranges), e.g.,

```
>> x = 1:4; x(2:end)
ans =
      2  3  4
```

`isreal`, e.g.,

```
>> ~isreal(i)
ans =
      1
```

`linspace` (linear space), e.g.,

```
>> linspace(0,2,3)
ans =
      0   1   2
```

`mod` (modulus), e.g.,

```
>> mod(4,2)
ans =
      0
```

`sum` (summation), e.g.,

```
>> sum(1:2)
ans =
      3
```

Problem 1 [12 points] *Symbolic Toolbox, MATLAB language syntax*

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

```
% [3 points] Problem 1a
>> clear; syms x; a=solve(x^2+4)
a =
```

```
% Problem 1b (1 point)
>> b = char('a' - 30)
b =
```

```
% Problem 1c (4 points)
>> ints = linspace(0,10,11);
>> c = sum(ints(mod(ints,2)==ones(1)))
c =
```

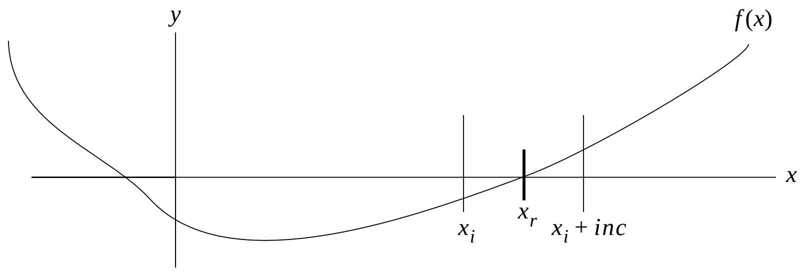
```
% Problem 1d (4 points)
>> data.s = { 'hi' };
>> data.s(2) = { data(1).s };
>> d = data.s{2}
d =
```

Problem 2 [42 points] *Numerical analysis, loops, recursion, functions, programming style*

Complete the code for the following function M-file called **`prob2.m`**. This code solves for the root of $f(x) = x^4 - 5x^2 - 22.5x - 1.5625 = 0$. The function returns a vector containing the positive and real root x_r, the corresponding value $f(x_r)$, and the number of iterations to compute x_r. Use the following algorithm, which is a hybrid of LHS/RHS and the bisection method:

- Assign problem data.
- Pick an initial root *x* and find $f(x)$, which is called **`lhs`**.
- While **`lhs`** does not meet tolerance and does not iterate "forever":
  - Check if the sign of **`lhs`** changes for the *next* value of the root. If the sign does change, divide the increment by a factor of 10. Otherwise, do not change the increment.
  - Compute the next value of x and $f(x)$.

For instance, the figure below demonstrates a value x_i. During the loop, the program would compute $x_i + inc$ and discover that the increment **`inc`** must be reduced because the root is between x_i and $x_i + inc$.



For full credit, your code must:
- avoid excessive iteration without using a **`break`**.
- use the provided subfunction **`rhs`**, provided at the bottom of the next page.
- provide commentary inside the box for the **`while`**-loop on the next page.

```
function stuff = prob2
%Solve for a root of the polynomial x^4 - c1*x^2 - c2*x - c3

%  Initialize values
   c1=5; c2=22.5; c3=1.5625;  % set constants in equation
   eps     = 0.001;           % tolerance
   inc     = 1;               % initial increment of root
   maxiter = 1000;            % maximum iterations allowed
   iters   = 0;               % iterations so far

%  Prompt user for initial value of the root $x$
%  Ensure $x$ is positive and real

   x = input('Enter initial value of root: ');

   while ________________________________________________

      x = input('That value is "evil". Re-enter initial value of root: ');

   end


%  Get initial value of $lhs$:
   __________ = ___________________________________________ ;
```

(Problem 2 continues on the next page.)

```
%  Iterate for the value of the root $x$.
%  Avoid excessive iteration by checking $iters$ with $maxiter$:

while ____________________________________________________________________________
```

```
end

%  Re-execute the function if excessive iteration was encountered:
   if _________________________________________

      disp('Too many iterations...restarting!');

      eval(_______________________) ; % Repeat the analysis
      _____________________________ ; % return without passing values
   else

%  Store final value of root, value of f(x) for the root, and iterations:

      _________ = ______________________________________ ; % array to return to user
   end

% Subfunction
function lhs = rhs(x,c1,c2,c3)
   %Compute left-hand side $lhs$ of equation.
   % Subfunction RHS computes the left-hand side of the given polynomial f(x)
   lhs = x^4 - c1*x^2 - c2*x - c3;
```

Problem 3 [40 points] *Functions, logical arrays, accumulation & conditional-update loops*

The following problems 3a, 3b, 3c, and 3d require you to manipulate sets. A ***set*** is defined as a collection of unique, or non-repeating, items. For MATLAB, you may consider a set as a row vector without duplicates. For instance, **`[3 2 1]`** is a set, whereas **`[1 3 2 1]`** is not. Hint: Each problem can be done separately.

3a) [21 points] Write a function **`vec2set.m`** that converts a vector **`v`** of numbers to a set of numbers, which is reassigned to **`v`**. An explanation for an algorithm we want you to use follows below. The algorithm:
- compares each element of **`v`** with the other elements of **`v`**.
- works from left to right with a loop.
- compares the current element of **`v`** (called the *current first-element*) with all remaining elements to the "right" (called **`rest_of_v`**).
- reassigns **`rest_of_v`** to be a vector with the duplicate elements removed from **`rest_of_v`**.
- reassigns **`v`** to be a vector that includes all previous first-elements, the current first-element, and the non-duplicate elements to analyze for the next iteration.

```
function v = vec2set(v)
%VEC2SET(V) converts a vector V of numbers into a set by removing duplicate values:

   ii = ___________; % current index value so far

   % Loop until index of current element exceeds length of set:

   while ________________________________________

      firstelem = ________________________________ ; % extract current first-element

      rest_of_v = ________________________________ ; % extract remaining elements

      % Extract elements from $rest_of_v$ that are not duplicates of $firstelem$:

      rest_of_v = ________________________________________________________________ ;

      % Reassign $v$ to a new vector that stores the previous first-elements, the
      % current first element, and the remaining elements that are not duplicates of
      % $firstelem$:

      _________ = ________________________________________________________________ ;

      _________ = ________________________________ ; % increment the current index

   end
```

3b) [4 points] Write a function **`set_union.m`** that takes the ***union*** between two sets **`x`** and **`y`**, denoted as $x \cup y$. The union of two sets is a set that contains all elements from both sets, but must still not contain any duplicates. Your code must use the **`vect2set`** function developed in Problem 3a – you assume that you got it right.

```
function result = set_union(x, y)
%SET_UNION(X,Y) returns the union of sets X and Y by combining all elements in a
%vector and removing duplicate items from that vector.
```

(Problem 3 continues on the next page.)

3c) [14 points] Complete the code for the function **set_intersect.m** which takes the ***intersection*** between two sets **x** and **y**, denoted as $x \cap y$. The intersection of two sets is a set that contains only elements contained in both sets. Working from left to right with **x**, compare each element of **x** with the set **y**. Using a logical array, you can determine if the current element of **x** appears at least once in **y**. Your code should store each intersecting element in an array **result** to return to the user.

```
function result = set_intersect(x,y)
%INTERSECT(X,Y) returns the intersection of sets X and Y by extracting elements
%common to both X and Y and disregarding all other elements.

%  Ensure that $x$ and $y$ are indeed sets
   x = vec2set(x); y = vec2set(y);

%  Initialize values
   ii = 1;         % current index
   result = [];    % intersection so far

%  Form intersection by comparing each element in $x$ with $y$:
   while __________________________________________ % check index with length of $x$

      % Determine if current element of $x$ appears at least once in $y$:

      if ____________________________________________

         result = ___________________________________ ; % store intersecting element

      end

      _______________________________________________ ; % increment counter

   end
```

3d) [7 points] Complete the M-file script **distribute.m** that proves a distributive law for sets:

$$x \cup (y \cap z) = (x \cup y) \cap (x \cup z).$$

Assume that **vec2set.m**, **set_union.m**, and **set_intersect.m** belong to a customized toolbox that your current MATLAB workspace can access. The script will report whether or not this theorem is true for the given **x**, **y**, and **z**.

```
%  distribute.m
%  Test one of the set distribution laws for the following vectors:
   x = [1 2 3 1]; y = [2 2 4]; z = [0 2 1 3];
%  Must first convert each vector into a set:
   x = vec2set(x); y = vec2set(y); z = vec2set(z);

%  Compute left- and right-hand sides of distribution law using $x$, $y$, and $z$:

   LHS = ____________________________________________________________________________

   RHS = ____________________________________________________________________________

if _____________________________

   disp('Success! The law worked. Hooray!');
else
   disp('Failure! The law failed. Bummer!');
end
```

Checklist: Congratulations! You reached the last page of Prelim 2. 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 points] Remember that bonus points do not count towards your core-point total! You will lose additional points from your *entire* CS100M bonus score for "inappropriate" language.

Bonus Problem 1 [2 points: 0.25 point per blank, rounded to nearest integer]: The ***first*** names of the eight CS100M teaching assistants are:

________________ ________________ ________________ ________________

________________ ________________ ________________ ________________

Bonus Problem 2 [2 points]:

```
>> data.s = { 'hi' };

>> data.s(2) = { data(1).s };

>> sum(data.s{2}{1}(1:2))
ans =
```

Bonus Problem 3 [2 points, Courtesy of NN]: What does **`blah.m`** do? See below:

```
function result = blah(x, y)
max_size = max(length(x), length(y));
x = [x, ones(1,max_size - length(x))*NaN];
y = [y, ones(1,max_size - length(y))*NaN];
result = sum(triu(repmat(x, [length(x), 1]) == repmat(y', [1, length(y)]))) == 1;
```

Bonus Problem 4 [3 points]: See next page.

CS100M Fall 2000 Mid-Semester Evaluation Form

By answering the following questions below, you will receive 3 bonus points on this exam. Do *not* answer these questions until you have finished the exam. How does this work? When you hand in your exam, the proctor will check to see if you answered a majority of these questions. If so, s/he will record the 3 points on the exam and then remove this sheet from the exam to maintain anonymity.

- We are curious about how students are reacting to the content of CS100M and the split between MATLAB and Java. How does the change from MATLAB to Java make you feel as of ***this week***? Answer 0 for the least and 5 for the most for each feeling, below. (At the end of the semester, we will ask you again to compare potential differences.)

Nervous: _____	(0 – you feel like a dew drop in the mist of morning; 5 – sirens are going off in your head)
Excited: _____	(0 – a root canal seems like more fun; 5 – you've already started sitting in on CS211)
Angry: _____	(0 – you use The Force only in self-defense; 5 – you've been seduced by the Dark Side)
Happy: _____	(0 – Java ushers in a bleak winter day; 5 – you're humming, "Happy days are here again!")
Apathetic: _____	(0 – this change is very important to you; 5 –- you couldn't care less)

- How do you feel about the pace of the course so far? Answer on a scale of 0 (too slow) to 5 (too fast) _____

- Name about three things that you feel would improve lecture. If you rarely (or never) attend lecture, please state why.

- Assuming you attend lecture, name about three things that you like about it (if any).

- Name about three things that you feel would improve section. If you rarely (or never) attend lecture, please state why.

- Assuming you attend section, name about three things that you like about it (if any).

- Do you have other comments/suggestions/critiques about CS100M? Use the back of this sheet if you need more space.