University of California, Berkeley — College of Engineering
Fall Semester 2005 — Professor R. Dibble and R. Horowitz

E77 Midterm Examination III

Monday November 21, 2005

Name :	
SID :	

Section: **1** **2** (Please circle your lecture section)

Please circle your Laboratory section: (where your exam will be returned)

#11: TuTh 8-10 #12: TuTh 10-12 #13: TuTh 12-2 #14: TuTh 2-4 #15: TuTh 4-6

#16: MW 8-10 #17: MW 10-12 #18: MW 2-4 #19: MW 4-6

Part	Points	Grade
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
TOTAL	70	

1. Write your name on each page.
2. Record your answers ONLY on the spaces provided.
3. You may not ask questions during the examination.
4. You may not leave the room before the exam ends.
5. Close book exam. 3 8.5" × 11" sheets (6 pages) of handwritten notes allowed.
6. No calculators or cell phones allowed. (Please turn cell phones off)

1 Part

(2) 1. Complete the following matlab function `roll_dice`, which simulates the rolling of two dice. Recall that the rolling of a single dice produces an integer outcome with value 1,2,3,4,5, or 6, and all outcomes have the same probability of occurrence.

```
function number = roll_dice
% This function simulates the rolling of two dice.
% Its output is the sum of the outcomes of the two dices

d1 = randperm(6);   % generate a random permutation
                    % of the integers 1 to 6
                    % e.g. randperm(6) might be [2 4 5 6 1 3]

d2 = ______________________________________________;

number = d1(1) + d2(2);
```

(8) 2. Suppose that you wish to determine the probability, `pr`, of getting either a sum of 2 or 7 (termed a "success") when you roll two dice.

Complete the following Matlab code intended to approximately determine this probability as the ratio of the number of successes to the total number of trials.

```
n = 0;          % number of ro1ls yielding either a 2 or a 7

N = 10000;      % total number of trials

for k = 1:N

    value = roll_dice;

    if ____________________________________________________

        n = ______________________________;

    end
end

pr = ____________________________;
```

2 Part

(10) Assign a **distinct** number to each element of `x` below, so that the function `interp1` returns the result shown.

```
>> clear all

>> X = [2 3 4 5];

>> Y = [2 4 3 5];          % Note that Y is not equal to X


>> x = [________________,________________];


>> y  = interp1(X, Y, x)

y =

    3.5000  3.5000
```

Hint: It may be helpful to plot the elements of `Y` versus the elements of `X`.

Help on function `interp1`

```
interp1 1-D interpolation (table lookup)
    yi = INTERP1(X,Y,xi) interpolates to find yi, the values of the
    underlying function Y at the points in the array xi using linear interpolation.
    X  and Y are vectors of length N.
```

3 Part

(10) Complete the code below so that it correctly implements the bisection algorithm:

```
function rt = bisection(fh, x1, x2, tol)

f1 = feval(fh, _________ );

f2 = feval(fh, x2);

if f1*f2 >= 0

      error('Incorrect braketing')

end

while abs(x1-x2)>tol

      x3 = 0.5*(x1+x2);

      ________ = feval(________,x3);

      if f1*f3 > 0

            x1 = x3;

            f1 = ____________;

      else

            x2 = ____________;
      end

end

rt = x1;
```

4 Part

(10) Complete the code of the function `zero_newton` below, which must implement Newton's algorithm to obtain a root of a `n`-th order polynomial, with coefficients given by the the `n+1` vector `p`, given a user-supplied initial guess `x0`.

Use the Matlab functions `polyval` and `polyder` to implement Newton's algorithm.

```
zero_newton syntax:        [x,r] = zero_newton(p,x0,tol,N)
where:
p:      is a 1-dimensional array  containing the polynomial coefficients
x0:     is the initial guess of the root
tol:    tolerance value
N:      maximum number of allowable iterations
x:      the root obtained by Newton's algorithm (if it converges)
r:      r = polyval(p,x) = p(1)*x^n + p(2)*x^(n-1) + ... + p(n+1)
```

Complete the function:

```
function [x,r] = zero_newton(p,x0,tol,N)

x = x0;
r = polyval(p,x);
k = polyder(p);

while (abs(r) > tol) _________ (N > 0)
    % update x (using Newton's algorithm), r and N

    x = _________________________________________;

    r = _________________________________________;

    N = _________________________________________;

end
```

Help on `polyder`: (from matlab)

`k = polyder(p)` returns vector `k` that represents the derivative of the polynomial represented by the vector `p`, for example:

```
>> k = polyder([ 1 3 2 3])
k = 3 6 2
```

5 Part

Write down what you think will be **the expected** (i.e. the most probable) output of the following Matlab commands

(2) 1.

```
>> clear all;
>> y =   2*rand(1e5,1)-1;   % rand: uniformly distributed
                            % random number generator
>> ybar = mean(y)
```

ybar = ______________________

(4) 2.

```
>>A = sum(y <= 0)
```

A = ______________________________________

(2) 3.

```
>> clear all;
>> y =   2*randn(1e5,1)+2;  % randn: normally distributed
                            % random number generator
                            % (Gaussian)
>> ybar = mean(y)
```

ybar = ______________________

(2) 4.

```
>> sigma = std(y)
```

sigma = ____

6 Part

Write the output of the following:

(2) 1.

```
>> a = 1e18; b = 3;
>> c = (a - a) + b , d = (a + b) - a
```

c = ________ d = ________

(2) 2.

```
>> clear all;
>> f = 1;
>> c = (f < (f + eps)) , d = (f < (f + eps/2))
```

c = ________ d = ________

(2) 3.

```
>> clear all;
>> f = 100;
>> c = (f < f*(1 + eps)) , d = (f < (f + eps))
```

c = ________ d = ________

(2) 4.

```
>> g = bin2dec('10011')
```

g = ________

(2) 5.

```
>> h = dec2bin(12)
```

h = ________

7 Part

(10) Complete the **recursive** MATLAB function `mfrac` below, which evaluates y from the following equation:

$$y = x(n) + \cfrac{1}{x(n-1) + \cfrac{1}{x(n-2) + \cfrac{1}{\cdots + \cfrac{1}{x(2) + \cfrac{1}{x(1)}}}}}$$

where x is an array of length n, containing positive elements.

```
function y = mfrac(x)

n = length(x);

if n > 1

   ____________________________________________;

else

   ____________________________________________;

end
```