Final examination revision - 202 CSc.

Question 1

Use MATLAB to evaluate the following expressions.

1. $1000(1 + 0.15/12)^{60}$
2. $(0.0000123+5.678\times10^{-3})\times0.4567\times10^{-4}$

Solution :

1.

```
1000*(1+0.15/12)^60
```

2.

```
(0.0000123+5.678*10^(-3))*0.4567*10^(-4)
```

Question 2

Set up a vector n with elements 1, 2, 3, 4, 5. Use MATLAB array operations on the vector n to set up the following four vectors, each with five elements:

a. 2, 4, 6, 8, 10

b. 1/2, 1, 3/2, 2, 5/2

c. 1, 1/2, 1/3, 1/4, 1/5

d. 1, $1/2^2$, $1/3^2$, $1/4^2$, $1/5^2$

Solution :

```
a = 1:5
b = a*2
c = a/2
d = a.\1
e = a.^2.\1
```

Question 3

Write MATLAB programs to find the following sums

(a) with for loops, and

(b) by vectorization.

1- 1/3 +1/5 -1/7 + 1/9 - ... -1/1003

Solution :

```
sign = -1;                          % with for loop
s = 0;
for i = 1:2:1003
 sign = -sign;
 s = s + sign/i;
end
disp(s);
```

```
a = 1:4:1001;                    % by vectorization
b = 3:4:1003;
s = sum(a.\1) - sum(b.\1);
```

Question 4

The probability density function of the random variable ***T*** described by the normal distribution is given as

$$f(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$$

1. Write a program to compute and display the values of ***T*** and f(t) over the range
   (-3,3) with step 0.01.
2. Plot a graph of *f(t)*, consider the following:
   - write the text '*The probability density function of the normal distribution* ' as a title on top of the graph.
   - label the *x*-axis as ***t***
   - label the *y*-axis as ***f(t)***.

Solution :

```
m = 0; % mean of probability distribution
s = 1; % standard deviation
t = -3 : 0.01 : 3; % random variable T
f = 1/(sqrt(2*pi)*s) * exp(-0.5*(t-m).^2/s^2);
plot(t, f),
title('The probability density function of the normal distribution');
xlabel('t'), ylabel('f(t)')
grid
disp([t' f']) %display a table
```

Question 5

It has been suggested that the population of the United States may be modeled by the formula

$$P(t) = \frac{197273000}{1 + e^{-0.03134(t-191325)}}$$

where *t* is the date in years.

Write a program to compute and display the population every *ten* years from 1790 to 2000. Plot a graph of the population against time

Solution :

```
t = 1790: 10: 2000; % date T in years
P = (1+exp(-0.03134 *(t-1913.25))).\ 197273000;
plot(t, P), title(' Model of the United states population ');
disp([t' P']) %display a table
```

Question 6

Write a script m-file to compute the solution of a quadratic equation. check whether *a* = 0, to prevent a division by zero.

Solution :

```
% Step 1: Start
format compact
disp(' Solution of Quadratic Equation ')
% Step 2: Input data
A = input(' Specify coefficients as follows: [a, b, c] = ');
a = A(1);b = A(2);c = A(3);
% Step 3: Evaluation of roots
if a==0 & b==0 & c==0
  disp(' Solution is indeterminate ')
elseif a==0 & b==0
  disp(' There is no solution ')
elseif a==0
   x = -c/b;
else
    if (b ^ 2 - 4*a*c == 0)
      x = –b / (2*a);
   elseif (b^ 2 – 4*a*c < 0)
     disp( 'Complex roots' )
   else
     x1 = (–b + sqrt(b^ 2 – 4*a*c)) / (2*a);
     x2 = (–b – sqrt(b^ 2 – 4*a*c)) / (2*a);
    end
end
% Step 4: Stop
```

Question 7

Write a script which inputs any three numbers (which may be equal) and displays the larger one with a suitable message, or if they are equal, displays a message to that effect.

Solution :

```
format compact
disp(' Find maximum number ')
% Step 1: Input data
A = input(' Enter three integers as follows [a,b,c]= ');
a = A(1);b = A(2);c = A(3);
% Step 3: Evaluation of roots
max =a;
if max==b & max==c
  disp(' all numbers are equal ')
else
   if max<b
      max=b; , end
   if max<c
      max=c; , end
end
disp(['Maximum number is: ',num2str(max)])
```

Question 8

Using *fix* and *rem* matlab commands write a script to convert seconds into hours, minutes and seconds. Try out your script on 10 000 seconds, which should convert to 2 hours 46 minutes and 40 seconds.

Solution:

```
t1 = 1000;            % first total of 10 000 seconds
hh = fix(t1/3600)     % hours
t2 = rem(t1,3600)     % second total of  1000 seconds
mm = fix(t2/60)       % minutes
ss =  rem(t2,60)      % seconds
```

Question 9

Design an algorithm (i.e. write the structure plan) for a machine which must give the correct amount of change from a RS100 note for any purchase costing less than RS 100. The plan must specify the number and type of notes 50, 20, 10 ,5 and 1 riyal in the change, and should in all cases give as few notes as possible.

Solution:

```
P = input(' enter purchase amount: ');   % purchase less than 100
S = 100 - P;              % change amount
D = [50 20 10 5 1];       % type of notes
N = [0 0 0 0 0];          % number of each type of notes
for i = 1:5
    while S >= D(i)
      N(i) = N(i) + 1;
      S =  S - D(i);
    end;
 end
 for i = 1:5
 disp(['Type: ', num2str(D(i)), ', number: ',num2str(N(i))]);
 end
```

Question 10

Write a script that plots the graph defined by

$$y(x) = \begin{cases} sin(x) & (sin(x) > 0) \\ 0 & (sin(x) \leq 0) \end{cases}$$

over the range 0 to 5π

Solution:

```
x = 0 : pi/20 : 5*pi;
y = sin(x);
y = y .* (y > 0); % set negative values of sin(x) to zero
plot(x, y)
```

Question 11

Write a script that simulates rolling a fair dice by generating a vector ***d*** of 2000 random integers in the range 1 to 6 and estimate the probability of throwing a six by dividing the number of sixes thrown by 2000.

Solution:

```
d = floor(6 * rand(1, 2000)) + 1  % Generate a random vector d in the range 1 to 6
n = sum( d == 6)   % Count the number of "sixes" thrown
p = n/2000     % Estimate the probability of throwing a six
```

Question 12

Set up any 3 × 3 matrix a. Write command-line statements to perform the following operations on a:

1. interchange columns 2 and 3;
2. add a fourth column (of 0's);

3. insert a row of 1's as the new second row of a (i.e. move the current second and third rows down);
4. remove the second column.

a = [1 2 3;4 5 6;7 8 9]
results in
a =

```
1   2   3
4   5   6
7   8   9
```

(1) Interchange columns 2 and 3.
a(:,[2 3]) = a(:,[3 2])
results in
a =

```
1   3   2
4   6   5
7   9   8
```

(2) Add a fourth column (of 0s).
a(3,4) = 0
results in
a =

```
1   3   2   0
4   6   5   0
7   9   8   0
```

(3) Insert a row of 1s as the new second row (i.e., move the current second and third rows down).
a([3 4],:) = a([2 3],:);
a(2,:) = ones
results in
a =

```
1   3   2   0
1   1   1   1
4   6   5   0
7   9   8   0
```

(4) Remove the second column.
a(:,2) = [ ]
results in
a =

```
1   2   0
1   1   1
4   5   0
7   8   0
```