NTNU
Kunnskap for en bedre verden

Department of Computer and Information Science

Final Examination in TDT4105 “Information Technology, Introduction”

Contact during the exam:	Rune Sætre	Mobil: +47 452 18103
	Anders Christensen	Mobil: +47 918 97181
	Nils Inge Rugsveen	Mobil: +47 958 76417

Exam date:	**2014-12-06**
Exam time (from-to):	**09:00 – 13:00**
Allow aids:	**Specified, simple calculator**

Other information:

The exam contains 4 problems. A percentage score is given to show how much each problem and sub problem counts when the exams are graded. Read through all the problems before you start solving them. Be smart and make good use of your time! If you feel the problems are not fully specified, please write your assumptions explicitly.

Answer briefly and clearly, and write so that the text is easy to read. If the text is ambiguous or longer than necessary, points will be deducted.

Language:	**English**
Number of pages:	**17 (including front-page, forms and appendix)**

Contents:

Controlled by:

Date Sign

Problem 1: Multiple Choice Questions (25%)

Use the two enclosed forms to solve this exercise (take one home). You can get a new form if you need it. Only one answer is completely correct. For each question, a correct answer counts 1 point. Wrong answer or more than one answer counts -1/2 point. No answer counts 0 points. You get no less than 0 points total for this problem.

1. What basic process activity do you find in a software development process that focuses on changing the software to meet changed customer and market requirements?
    a. Software specification.
    b. Software design and implementation.
    c. Software validation.
    d. Software evolution.
2. What is an analogue signal?
    a. A continuous signal where the variable characteristics are given by a discrete function, as it gives values from a defined and limited range.
    b. A continuous signal where the variable characteristics (like amplitude or frequency) represent the information being transferred.
    c. A discrete signal being represented by zeros and ones.
    d. A combination of alternative a and b.
3. What type of loop is guaranteed to be executed at least once?
    a. pretest loop.
    b. posttest loop.
    c. both types.
    d. neither of the types.
4. About how many times faster is a 1 GHz-processor compared to one on 2 MHz?
    a. Half as fast.
    b. About the same.
    c. Twice as fast.
    d. 500 times as fast.
5. What does the Nyquist rule say?
    a. That the sampling rate of audio must be at least twice as fast as the fastest frequency.
    b. That audio over 20000Hz cannot be recognized by the human ear.
    c. That loss-free compression is not possible for audio.
    d. That audio data can be compressed with a maximum factor of 2*pi.
6. What software process model should be chosen for a project where a totally new system will be developed where there are no existing software components, and the customer is uncertain how the system should be?
    a. The waterfall model.
    b. Incremental development.
    c. Reuse-oriented software engineering.
    d. The ocean model.
7. What is the purpose of a parity bit for digital signals?
    a. Tells where the message should be sent.
    b. Make messages faster to transfer (compression).
    c. Contributes to detect errors in digital signals.
    d. Encrypt signals to make the transfer of data more secure.

8. The complexity of insertion sort is
    a. Θ(n).
    b. Θ(n log n).
    c. Θ(n^2).
    d. Θ(2n).
9. A modern processor is typically made up of millions of tiny…
    a. Diodes.
    b. Magnets.
    c. Transistors.
    d. Capacitors.
10. A byte of memory in a computer can store how much?
    a. 16 bits.
    b. 8 floating point numbers.
    c. Four ASCII-characters.
    d. An integer between 0 and 255.
11. Which of the following is a known advantage of the waterfall model?
    a. Adapts to user requirements that changes during the project.
    b. Makes the process visible and easier to monitor for the project manager.
    c. Produces early versions of the system to the customer.
    d. Opens for continuous feedback from users of the system.
12. Morse code represents letters as sequences of dots and lines, which are…
    a. Same length for all letters in the alphabet.
    b. Shorter for letters early in the alphabet, longer for letters at the end.
    c. Shorter for vowels, longer for consonants.
    d. Shorter for letters that are frequent in normal text, longer for letters that are not used as much.
13. Which of the following is a correct definition of algorithm according to the textbook? "An algorithm is an ordered set of…"
    a. "…unambiguous, executable steps that defines a terminating process".
    b. "…unambiguous, efficient steps that defines an executable process".
    c. "…well-formed, efficient steps that defines a terminating process".
    d. "…well-formed, efficient steps that defines an efficient process".
14. A computer runs an eternal loop named
    a. The Natural Cycle.
    b. The Fetch/Execute Cycle
    c. The Eternal Cycle.
    d. The Computational Cycle.
15. What is the correct binary representation of 'NTNU' in 8-bits ASCII?
    a. `01001110 01010100 01001110 01010101`.
    b. `01100001 01100100 01110011 01100110`.
    c. `01101110 01110101 01110100 01001110`.
    d. `01100010 01010101 01010010 01010000`.

16. For what case below is reuse-oriented software engineering most useful?
    a. When there exist available software that can do the system’s tasks.
    b. When developing software for systems to manage recycling of trash or similar.
    c. When ideas from previous projects are being reused.
    d. When software developers and designers from previous projects are being reused.
17. What is ISP an abbreviation for?
    a. Internet Service Provider.
    b. Information Security Protocol.
    c. Internet Security Protocol.
    d. Information Super Pool.
18. The complexity of a binary search is
    a. Θ(n) if the list is sorted and Θ(n log n) if it is not sorted.
    b. Θ(log n) if the list is sorted and Θ(2 log n) if it is not sorted.
    c. Θ(log n) if the list is sorted and Θ(n) if it is not sorted
    d. Θ(log n) if the list is sorted. Binary search does not work if the list is not sorted.
19. RAM
    a. Remembers all the values after the power is shut down.
    b. Is always divided into blocks of 1 kilobyte.
    c. Means Random Access Memory.
    d. Can safely be removed without causing the computer to stop.
20. What is RGB an abbreviation for?
    a. Red, Green, Blue.
    b. Readable Graphics Byte.
    c. Raster Grayscale Balance.
    d. Real-time GPU Backlog.
21. What is the name of the activity with focus on identifying the overall structure of the system including its sub-systems?
    a. Main design.
    b. Architectural design.
    c. Interface design.
    d. Component design.
22. MODEM is an abbreviation for
    a. MOdulator / DEModulator.
    b. Massive Online Digital Electric Messaging.
    c. MOnitored Data EMission.
    d. Mapping Of Digital Electronic Mail.
23. ASCII-code represents the letters A to Z as 0/1 sequences which are
    a. The same length for these letters in the alphabet.
    b. Shorter for letters early in the alphabet, longer for letters late in the alphabet.
    c. Shorter for vowels, longer for consonants.
    d. Shorter for letters that are frequently used in normal text, longer for letters that are not used that often.

24. A network to connect computers and devices in a limited area as in an office, a building or a house is described by the abbreviation:
    a. LAN.
    b. MAN.
    c. PAN.
    d. WAN.

25. VPN (Virtual Private Network) can provide the receiver the impression that a travelling employee's PC is directly connected to the company's network as the messages from this PC
    a. Is placed in an encrypted data packet for external transfer.
    b. Is sent with latency (time delay).
    c. Is sent very fast, with high priority.
    d. Is sent with a false from-address that includes a virus.

Problem 2 Programming Parachuting (25%)

(In this problem, it can be beneficial to call functions you have created in previous sub-problems. Even if you have not solved a previous sub-problem, you can call a function defined in the sub-problem with the assumptions that it works as specified.)

NTNU fallskjermklubb (NTNU-FSK) (parachuting club) needs your help to create a new training- and administration system. They have asked your company (ITGK) for help, and you have got the job to program functions as described in the sub-problems below.
In this problem, a simplified version of physical laws are used:

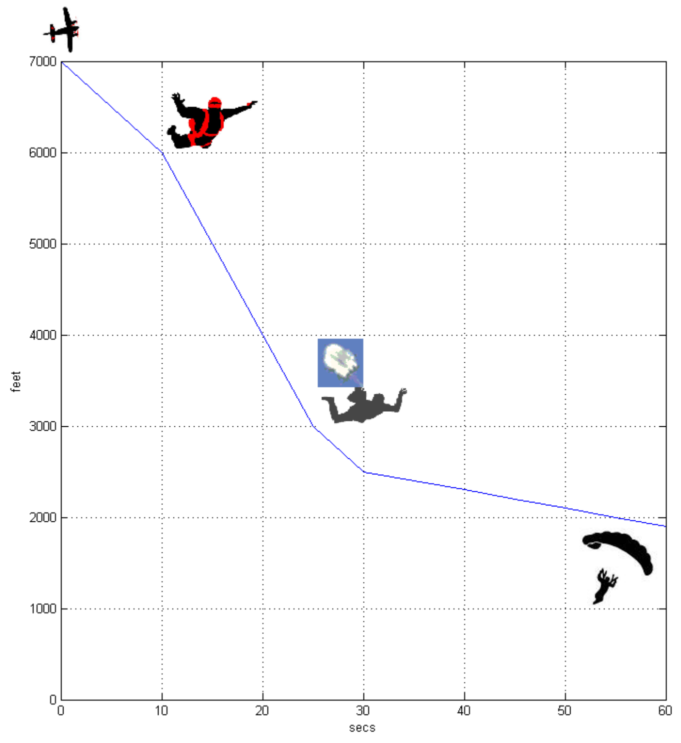

Figure 1. Skydive from 7000 feet.

A skydiver falls (with a constant/average speed) 100 feet per second the first 10 seconds, then with a constant top speed 200 feet per second until the parachute must be opened at the altitude of 3000 feet (see figure 1). If you jump against normal under 3000 feet, the parachute must be opened immediately (after 0 seconds).

NTNU-FSK's member database is stored in the file 'members.txt', in the following format:

NAME ; ID ; WEIGHT ; PARSIZE.
Example of the content of the file:

```
Frank Stank;D-49334;75;120
Bjarne Stor;C-49335;95;150
Dumbo Ear;D-50105;450;750
Peter Pan;A-12345;30;100
```

Problem 2a (5%)

Create the function `inputPerson` that reads in name, ID, weight and parachute size from the keyboard, and returns a struct `person` with the correct values stored in the following fields: name, id, weight, size. Note that name and ID are text strings, while weight and parachute size are integers. You can assume that the user enters legal values so you do not need to do any error handling.

Example of execution (underlined text is written by the user):

```
>> person = inputPerson()
Name: Fredrik Olsen
ID: B-77777
Kg: 80
Size: 240
person =
       name: 'Fredrik Olsen'
         id: 'B-77777'
     weight: 80
       size: 240
>>
```

Problem 2b (5%)
Create the function `readDbFile` that reads the whole member database into a struct vector `db` with the following fields: `name`, `id`, `weight` and `size` (see the description above). `name` and `id` are text strings, while the `weight` and `size` are integers. You can assume that the file exists, that there are no problems opening/closing the file, and that the file does not contain any blank or illegal lines. The function takes `filename` as an input parameter and returns the `db`.
Example of execution:

```
>> db = readDbFile( 'members.txt' )
db = 1x4 struct array with fields:
    name
    id
    weight
    size
>> db(1)
ans =
      name: 'Frank Stank'
        id: 'D-49334'
    weight: 75
      size: 120
>>
```

Problem 2c (5%)
Create the function `printMemberList` that prints the following header and the content of the struct vector `db` (described above) to the screen in the following format:

NAVN (15 characters allocated) ID-NR (9 characters allocated) VEKT (5 characters allocated) kg. SKJERMSTØRRELSE (4 characters allocated) kvadratfot

You can assume that the database does not contain any data that does not fit into the allocated number of characters.

Input `db`, no return values.
Example of execution, given that `db` contains the table shown in Problem 2b:

```
>> db = readDbFile('members.txt');
>> printMemberList( db )
           NAVN    ID-NR VEKT kg. SKJERMSTØRRELSE
    Frank Stank  D-49334   75 kg.  120 kvadratfot
    Bjarne Stor  C-49335   95 kg.  150 kvadratfot
      Dumbo Ear  D-50105  450 kg.  750 kvadratfot
      Peter Pan  A-12345   30 kg.  100 kvadratfot
>>
```

Problem 2d (5%)
Create the function `addPerson` with the input parameter `filename` (filename for the database). The function shall ask the user to input information about a person described by name, ID, weight, and parachute size, and store this information in the variable `person`. The function will then read the database stored in the file `filename` into the struct-vector `db` as described in Problem 2b. The information about the new person will then be added to `db` and to the file `filename`. Use correct format: see « Example of the content of the file» close to Figure 1 (page 6). The return value `db` must contain the updated database.

You do not have to write code for exception handling related to reading/writing files.
Example of execution: (everything underlined is information entered by the user, and the changes as result of executing the function are in **bold**):

```
>> db = addPerson( 'members.txt' )
Name: Santa Klaus
ID: H-12345
KG: 155
Size: 380
db =
1x5 struct array with fields:
    name
    id
    weight
    size
>> db(5)
ans =
     name: 'Santa Klaus'
       id: 'H-12345'
   weight: 155
     size: 380
>>
```

Problem 2e (5%)
For a skydiver it is very important to be aware of how many seconds he or she can wait before the parachute must be opened (see Figure 1). Create the function `feet2seconds` that computes how many second it takes to fall from a given altitude (given in feet) down to 3000 feet (input parameter `feet`, and return value `seconds`). Use the information given in the beginning of Problem 2 (explanations to Figure 1) to compute the correct time. If the number of feet is below 3000, the function shall return 0.

Examples of execution:

```
>> feet2seconds( 12500 )
ans =   52.5000
>> feet2seconds( 7000 )
ans =    25
>> feet2seconds( 2000 )
ans =     0
>>
```

Problem 3 Programming Weather Station (30%)

You must here manage data from a weather station over a large amount of days. The data are stored as float numbers in a table named `weatherData`. Every row represent measured data for *one day* and has three elements of different metrics. The three types of measured data are maximum temperature, minimum temperature, and amount of rain (precipitation). We refer to the days in such a way that the data from the first row is for day number 1, the data from the second row is from day number 2 etc.
Examples of printout from rows in `weatherData` could be:

```
>> weatherData
ans =
   12.0000    2.4000    8.2000
    6.1000    0.6000   11.9000
    8.3000   -3.5000    0.0000
   11.6000   -5.2000    0.0000
   15.3000    2.8000   14.3000
>>
```

Problem 3a (10%)

Write the function `weatherStats` that takes `weatherData` as input parameter. The function will go through the data, and based on them write a summary as shown below. This means that the function will print the number of days in the period, the total amount of rain (precipitation), the lowest and the highest temperature together with the day for these temperatures.

Example on execution for the data shown in the gray box above:

```
>> weatherStats(weatherData)
There are 5 days in the period.
The highest temperature was 15.3 on day number 5.
The lowest temperature was -5.2 on day number 4.
There was a total of 34.4 rain in the period.
>>
```

Problem 3b (10%)

Write the function `coldestThreeDays` that takes as input parameter `weatherData` (as defined above). The function should find the period of the three days in a row with the lowest average minimum temperature. It shall return the number of the first day of this three-day period. If there is more than one three-day period with the same lowest average minimum temperature, the function should return the last of these periods.
Example of execution of this function with `weatherData` as defined earlier in the problem:

```
>> coldestThreeDays(weatherData)
ans =
    2
>>
```

Problem 3c (10%)
The weather station has just reported data for yet another day. The data comes as a text string stored in the variable `extraData` formatted as shown below:

```
>> extraData = 'max=23.5, min=9.3, 5.1mm';
>>
```

Write the function `addNewDay` that takes `extraData` and `weatherData` as input parameters, and returns a new version of `weatherData` updated with the new data at the end of the table.

Example of execution (changes are shown in **bold**):

```
>> updated = addNewDay( extraData, weatherData );
>> updated(end-2:end,:)
ans =
   11.6000   -5.2000    0.0000
   15.3000    2.8000   14.3000
   23.5000    9.3000    5.1000
>>
```

Problem 4 Understanding Code (20%)

Problem 4a (5%)

What is returned when executing the function `myst([1,2,3,3,2,1])` with code as shown below? (3 %)
Describe with <u>one sentence</u> what the function `myst` does? (2 %)

```
function out = myst(A)
  out = 0;
  L = length(A);
  if ( L > 1 )
    B = A(1) * A(L);
    out = B + myst( A(2:L-1) );
  end %if
end %func
```

Problem 4b (5%)

What is printed to screen when the function `myst_b()` is executed? (3 %)
Describe with <u>one sentence</u> what the function `mystery` does? (2 %)

```
function myst_b( )
      disp( mystery(4) )
end

function out = mystery( W )
    %Creating a Matrix with W x W zeros
    table = zeros( W );
    for a = 1:W
        table(a,a)=1;
        b=0;
        while a-b > 1 && a+b < W
            b = b+1;
            table( a-b, a+b ) = 1;
            table( a+b, a-b ) = 1;
        end
    end
    out = table;
end
```

Problem 4c (5%)

What is returned when executing the function `myst_c('RBHOOASDUEØGNGEBLSOIURMNGTD')` with the code as shown below? (3 %)
Describe with <u>one sentence</u> what the function `myst_c()` does? (2 %)

```
function B = myst_c(A)
  B='';
  for x = 1:3:length(A)
    B = [B A(x)];
  end %for
end %func
```

Problem 4d (5%)
A program to train middle-school students in math needs a function to test the correct nesting of parentheses in expressions where three different types of parentheses are allowed. The function does not need to test that the expression makes sense, apart from correct sequence of parentheses and correct balance between start and stop parentheses. In the code below, there are also three calls to the function to test if it works correctly. All the code is shown below (the line numbers to the left are not a part of the code, but are included so you can easier refer to specific lines of code in your answer):

```
1 function check_main( )
2     A = check( '{a+4*[b-2*(c+5)]/11}' ) %  should be  true
3     B = check( '{a+4*[b-2*(c+5])/11}' ) %  should be  false
4     C = check( '{a+4*[b-2*(c+5])/11}}' ) % should be  false
5 end %function check_main

6 function out = check( expression )
7    PAREN = '([{)]}';
8    parenthesis_list = [];
9    for char = expression
10        if find( char == PAREN(1:3) ) %start-parenthesis found
11            %add corresponding end-parenthesis to the back of the list!
12            parenthesis_list =  [ parenthesis_list...
13                                      PAREN( find( PAREN == char )+3 ) ];
14        elseif find( char == PAREN(4:6) ) % end-parenthesis found
15            if find( char == parenthesis_list ) % matched start/end-parenthesis
16                parenthesis_list( char == parenthesis_list ) = []; %remove it
17            else
18                out = false;
19                return
20            end
21        end
22    end
23    out = parenthesis_list;
24 end
```

As the comments in check_main shows, the results should have been true (1), false (0) and false(0). However, when the `main`-function is executed, we get the result:

```
>> check_main()
A =   Empty string: 1-by-0
B =   Empty string: 1-by-0
C =     0
```

There are <u>two errors</u> in the code:

1) The function returns `Empty string: 1-by-0` instead of a Boolean value for both A and B. If this error is corrected, the execution of the program will reveal the second error:

```
>> check_main()
A =     1
B =     1
C =     0
```

2) as shown above: The function now returns true (1) for an expression that should have been false (0), as in line B above (in **bold**).

<u>**Question:** Explain which lines of code that causes error 1) and error 2), and what the easiest way to correct them are. In both cases, it should be possible to fix the error by only changing existing code, so it is not necessary to add new lines of code.</u>

Appendix: Some useful functions

FIX Round towards zero.

FIX(X) rounds the elements of X to the nearest integers towards zero.

FLOOR Round towards minus infinity.

FLOOR(X) rounds the elements of X to the nearest integers towards minus infinity.

FCLOSE Close file.

ST = FCLOSE(FID) closes the file associated with file identifier FID, which is an integer value obtained from an earlier call to FOPEN. FCLOSE returns 0 if successful or -1 if not.

FEOF Test for end-of-file.

ST = FEOF(FID) returns 1 if the end-of-file indicator for the file with file identifier FID has been set, and 0 otherwise.

The end-of-file indicator is set when a read operation on the file associated with the FID attempts to read past the end of the file.

FGETL Read line from file, discard newline character.

TLINE = FGETL(FID) returns the next line of a file associated with file identifier FID as a MATLAB string. The line terminator is NOT included. Use FGETS to get the next line with the line terminator INCLUDED. If just an end-of-file is encountered, -1 is returned.

FIND Returns the linear indexes of non-zero elements in a matrix.

FIND([0 1 0 1 0]) returns [2 4]. If the first parameter has more than one row, a column vector containing the linear indexes of non-zero elements are returned. An optional second parameter set the maximum number of indexes to return.

FOPEN Open file.

FID = FOPEN(FILENAME,PERMISSION) opens the file FILENAME in the mode specified by PERMISSION:

'r'	open file for reading
'w'	open file for writing; discard existing contents
'a'	open or create file for writing; append data to end of file
'r+'	open (do not create) file for reading and writing
'w+'	open or create file for reading and writing; discard existing contents
'a+'	open or create file for reading and writing; append data to end of file

FPRINTF Write formatted data to file.

COUNT = FPRINTF(FID,FORMAT,A,...) formats the data in the real part of array A (and in any additional array arguments), under control of the specified FORMAT string, and writes it to the file associated with file identifier FID. COUNT is the number of bytes successfully written. FID is an integer file identifier obtained from FOPEN. It can also be 1 for standard output (the screen) or 2 for standard error. If FID is omitted, output goes to the screen.

FORMAT is a string containing ordinary characters and/or C language conversion specifications. Conversion specifications involve the character %, optional flags, optional width and precision fields, optional subtype specifier, and conversion characters d, i, o, u, x, X, f, e, E, g, G, c, and s.

The special formats \n,\r,\t,\b,\f can be used to produce linefeed, carriage return, tab, backspace, and formfeed characters respectively. Use \\ to produce a backslash character and %% to produce the percent character.

INPUT Read a value from the keyboard and into a variable

ANSWER=INPUT(STR) prints STR as a prompt, reads a number and assigns it to ANSWER. If character string are to be read, use the optional second parameter ‘s’.

ISEMPTY - Determine whether array is empty

This MATLAB function returns logical 1 (true) if A is an empty array and logical 0 (false) otherwise.

TF = isempty(A)

LENGTH The length of vector.
LENGTH(X) returns the length of vector X. It is equivalent to MAX(SIZE(X)) for non-empty arrays and 0 for empty ones.

MAX finds the highest element in a vector, or the highest element in each column of a matrix.

MIN finds the lowest element in a vector, or the lowest element in each column of a matrix.

MOD Modulus after division.
MOD(x,y) is x - n.*y where n = floor(x./y) if y ~= 0.

RANDI Pseudorandom integers from a uniform discrete distribution.
R = RANDI(IMAX,N) returns an N-by-N matrix containing pseudorandom integer values drawn from the discrete uniform distribution on 1:IMAX.
RANDI(IMAX,M,N) or RANDI(IMAX,[M,N]) returns an M-by-N matrix.

REM Remainder after division.
REM(x,y) is x - n.*y where n = fix(x./y) if y ~= 0.

SIZE The size of array.
D = SIZE(X), for M-by-N matrix X, returns the two-element row vector
D = [M,N] containing the number of rows and columns in the matrix.

SQRT Square root.
SQRT(X) is the square root of the elements of X.

SSCANF Extracts values from a string according to a format string. Opposite of FPRINTF.
A=SSCANF('12/11-2014','%d/%d-%d') returns a column vector containing the values
12, 11, and 2014.

STRSPLIT Splits the first (string) parameter into a cell array of substrings, according to the delimiter string given as the second parameter. STRSPLIT('one, two, three', ', ') results in {'one', 'two', 'three'}. Multiple alternative delimiters can be specified using a cell array as the second parameter.

STRTOK separates the first token of a string from the rest of that string.
[TOKEN, REST]=STRTOK(' first second', DELIM) sets TOKEN to 'first' and REST to ' second'. The optional parameter DELIM contains a list of delimiter characters – where the space character is default. Any delimiter characters before the first token are ignored.

STR2NUM Convert string matrix to numeric array.
X = STR2NUM(S) converts a character array representation of a matrix of numbers to a numeric matrix. For example, S=['12'; '34']   str2num(S) => [12; 34]

SUM The sum of elements.
S = SUM(X) is the sum of the elements of the vector X. If X is a matrix, S is a row vector with the sum over each column.

Answer Form for Multiple Choice Questions

Candidate number:______________________ Program: ______________________

Course code: ______________________ Date: ______________________

Total no. of pages:______________________ Page: ______________________

Problem	***A***	***B***	***C***	***D***
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				
1.21				
1.22				
1.23				
1.24				
1.25				

This page is on purpose empty!

Answer Form for Multiple Choice Questions

Candidate number:	______________________	Program:	______________________
Course code:	______________________	Date:	______________________
Total no. of pages:	______________________	Page:	______________________

Problem	***A***	***B***	***C***	***D***
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				
1.21				
1.22				
1.23				
1.24				
1.25				