NTNU
Kunnskap for en bedre verden

Department for Computer and Information Science

Continuation examination in TDT4110 Information Technology - Introduction

Contact during the exam:	Alf Inge Wang	Mobil: +47 922 89577
	Guttorm Sindre	Mobil: +47 944 30245
	Terje Rydland	Mobil: +47 957 73463

Exam date:	**2016-08-12**
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:	**19 (including front-page, forms and appendix)**

Content:

- Problem 1: Multiple Choice Questions (25%)
- Problem 2: Programming: Binary coding (25%)
- Problem 3: Programming: Sports for all (30%)
- Problem 4: Understanding code (20%)
- Appendix: Useful functions
- Forms for answering multiple choice questions (2 forms)

Controlled by:

22 juni 2016	Guttorm Sindre
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. A byte contains how many bits?
   a. 1
   b. 8
   c. 16
   d. 32

2. What is a test where you only consider inputs and outputs named?
   a. White box test.
   b. IO-test (Input Output test).
   c. Black box test.
   d. Integration test.

3. About how many devices are connected to the Internet today?
   a. Half as many as the population of the Earth.
   b. As many as the population of the Earth.
   c. More than three times as many as the population of the Earth.
   d. Impossible to answer.

4. What is a transistor?
   a. The unit that transforms 220V AC to DC, which can be used by the various part (CPU, audio card, graphics card, hard drive, RAM etc.) in a computer.
   b. A circuit board which connected all the parts of a computer, such as CPU, memory, audio card, graphics card, RAM etc.
   c. A switch that current can go through or not, which can be changed by electric current.
   d. The algorithm to transform physical audio signals to digital representation of 0s and 1s.

5. How do you represent the decimal number 321 as a binary number?
   a. `1100101`
   b. `11001001`
   c. `100101101`
   d. `101000001`

6. What is the purpose of a DAC?
   a. Convert from analogue to digital signal.
   b. Convert from digital to analogue signal.
   c. Compress a digital signal.
   d. Increase the sampling frequency.

7. What do we get with the same amount of red, green and blue in a pixel at a screen?
   a. Black, white or gray-tones.
   b. Cyan, magenta and yellow.
   c. Brown.
   d. Violet.

8. What is meant by 'sampling rate' in relation to digitalization of audio?
   a. How frequent the sound is measured.
   b. The frequency of the sound to be digitalized.
   c. The accuracy/number of bits of the measured value.
   d. The maximum audio volume that can be digitalized.

9. What is RGB?
   a. Random GB – the memory in a computer.
   b. A color model for reflected light, which shows how all the primary colors can be created.
   c. A color model for emitted light, which shows how all the primary colors can be created.
   d. Red, Grey and Blue – the primary colors of reflected light.

10. What is booting of a computer?
   a. The process of starting up a computer.
   b. Change the system preferences on a computer.
   c. To test if a computer has been infected by a virus (a boot)
   d. To kick (boot) a computer when it does not do what it is suppose to do.

11. What does the term ”system engineering” describe according to the textbook?
   a. ”System engineering” focuses on the underlying theories and methods, which makes computer systems.
   b. ”System engineering” focuses on practical problems with producing software.
   c. ”System engineering” includes all the aspects of development and evolution of complex systems where software plays an important part.
   d. ”System engineering” and ”Software engineering” is the same.

12. Which if these units are usually involved in the “Fetch/Execute Cycle”?
   a. ALU, CU (Control Unit), RAM.
   b. ROM, ALU, RAM.
   c. CU, RAM, ROM.
   d. OS, ALU, CU.

13. What is the main focus of the activity architectural design?
   a. Define interfaces between system components.
   b. Design how each system component should work.
   c. Design the data structures for the whole system.
   d. Identify the overall structure of the system, the overall components, and how they are structured and connected.

14. Regarding storing and transfer of data, they are respectively measured as…
    a. Bits (storing) and Bits pr. Second (transfer).
    b. Bits (storing) and Byte pr. Second (transfer).
    c. Byte (storing) and Byte pr. Second (transfer).
    d. Byte (storing) and Bits pr. Second (transfer).

15. A disadvantage with incremental development can be:
    a. All the requirements must be in place before design and implementation of the system can start.
    b. Makes it harder to test the system for bugs.
    c. Only fits for large projects.
    d. The architecture (structure) to the system has a tendency of degrading for every increment.

16. We can transfer more and more information through the air by…
    a. Use older stable encoders.
    b. Switch to lower frequencies.
    c. Exploit the interference, which occur when two receivers are in the same location.
    d. Increase the bandwidth.

17. Regarding the address fields in old (version 4) and new IP-version (version 6), the following is true…
    a. They use the same amount of bits.
    b. The old version did not have enough bits to address all the computers on the Internet.
    c. The new version uses twice as many bits.
    d. The new version manages to use half as many bits.

18. What is software evolution?
    a. Software must be changed due to changes in the execution environment.
    b. Run software on faster computers.
    c. Software which increasingly get bigger and faster.
    d. Software that gets more and more intelligent and flexible.

19. Assume that we have a two dimensional list (list of lists) with names, e.g. `lister = [ ['Anh, Ine', 'By, Ken', …], ['By, Ken', 'Cox, Jo', …], …]` Here, is every inner list alphabetically sorted by last name without duplicates, but the same name can exist in several of the inner lists. We want the function `antall_n(lister, navn)` that returns the number of list a name exist in. Here is pseudo code for such a solution, consisting of two functions where one – `antall_n( )` – calls the other named `antall( )`.

```
function antall_n (lister, navn):
      ant ← 0
      let liste_n gi from first to last element in lister:
            ant ←ant + antall(liste_n, navn) # function from pseudo code
            # for antall() below, will give 0 or 1
      returner ant
```

where `antall(liste, navn)` is a function, which returns an integer with how many times the name is found in a list like `liste =  [ 'Jo Å', 'Geir Li', 'Ine By']`

```
function antall (liste, navn):
      antall ← 0
      let n go from first to last element in liste:
            if n == navn:
                        antall ←antall + 1
      return antall
```

The complexity of this solution will then be:
a. Θ(n)
b. Θ(n log n)
c. Θ(n^2)
d. Θ(n^3)

20. Instead of using the function `antall()` within the function `antall_n`, we can use binary search, meaning that we exchange `ant ←ant + antall(liste_n, navn)` in the pseudo code with `ant ←ant + bin_search(liste_n, navn)` where we can assume that `bin_search` in this case returns 1 if `navn` is found in `liste_n` and 0 if not. The complexity of `antall_n` will then be:
a. Θ(n)
b. Θ(n log n)
c. Θ(n^2)
d. Θ(n^3)

Problem 2 Programming binary coding (25%)

You can assume that all functions receive valid arguments (input values). You can use functions from sub-problems even if you have not solved this sub-problem.
In this problem, you will create functions to read a text file with binary code and transform this to characters coded in own coding set for symbols and letters, and store the results to a text file.

Problem 2a (5%)

Create the function `load_bin` with the input parameter `filename`, which is the name of the file to be loaded. The function will read all content in the file and return the content as a text string without white spaces (return (new line) or space). The file the function should read from is a text file with binary digits (0 and 1). If the file does not exist or cannot be opened, the function will return an empty string as well as print the following error message to the screen: "Error: Could not open file <filename>", where <filename> is the name of the file.

Example on execution of the file "binary-file.txt" which contains the following:

```
0100010
001
1001010101
11011
```

```
>>>
>>> load_bin("binary-file.txt")
'0100010001100101010111011'
>>>
```

Example on execution of the file "binary-fill.txt", which does not exist:

```
>>>
>>> load_bin("binary-fill.txt")
Error: Could not open file binary-fill.txt
''
>>>
```

Problem 2b (5%)

Create the function `bin_to_dec` with the input parameter `binary`, which is a text string of unknown size consisting of binary digits (text string with 0 and 1s). The function shall return an integer (decimal system), which corresponds to the binary number given by the string `binary`. This problem ***shall not be solved*** by built-in functions to translate binary numbers to integers.

Example on use of the function:

```
>>>
>>> bin_to_dec("101")
5
>>> bin_to_dec("11111111")
255
>>> bin_to_dec("10000000000")
1024
>>>
```

Problem 2c (4%)
Create the function `dec_to_char` with the input parameter `dec`, which is an integer with a value between 0 and 31. The function shall return a symbol or letter corresponding to the value of `dec`:

- If `dec` has the value 0, the character" " (space) shall be returned
- If `dec` has the value 1, the character "," (comma) shall be returned
- If `dec` has the value 2, the character "." (period) shall be returned
- If `dec` has a value between 3 and 31, a capital letter in the Norwegian alphabet shall be returned, where `dec=3` gives `"A", dec=4` gives `"B"`, all the way up to `dec=31` gives the letter `"Å"`.
- For all other values of `dec` the function should return an empty string.

Example of use of the function:
```
>>> dec_to_char(0)
' '
>>> dec_to_char(1)
','
>>> dec_to_char(2)
'.'
>>> dec_to_char(3)
'A'
>>> dec_to_char(31)
'Å'
>>>
```

Problem 2d (4%)
Create the function `bin_to_txt` with the input parameter `binstring`, which is a string of unknown length containing binary digits (string of 0 and 1s). The function shall return a text string containing letters and characters coded according problem 2c where every character is represented by 5 bits. The input parameter `binstring` will always be of a multiply of five.

Example of use of the function:
```
>>>
>>> bin_to_txt("00010")
'.'
>>> bin_to_txt("00011")
'A'
>>> bin_to_txt("101000010001101")
'RBK'
>>>
```

Problem 2e (7%)
Create the function `main` without any parameters. The function shall do the following:

1. Write the text ”Binary-to-text converter” to the screen
2. Ask the user about the name of the file to be loaded (text file containing binary digits) and store the filename in the variable `b_file`.
3. Load the file `b_file` and store the content of the file to the variable `b_string` (string of binary digits
4. Translate the string of binary digits to a text of characters and letters, and store the content in the variable `txt`.
5. Ask the user about the name of the file where the results will be stored, and store the file name in the variable `t_file`.
6. Write the content of the variable `txt` to the file with filename given by the variable `t_file`.
7. Print the following to the user:

”`<b_file>` has been converted and saved to `<t_file>`”

If the function get any trouble of writing to the file, the following error message should be printed to screen: ”Error: Could not write to file `<t_file>`”. (What has been specified between the < > in these printouts shall be replaced by the content of the variables as shown below).

Example on execution of the function:

```
>>>
>>> main()
Binary-to-text converter
Name of binary file to load from: binary.txt
Name of text file to save to: out.txt
binary.txt has been converted and saved to out.txt
>>>
```

The content of binary.txt :

```
01100101111010110110
010111000000000
00100010110011100100
001111010000000
01000011101000110001
1010000000
00101011100011100011
10000010111000001001
00010
```

The content of out.txt :

```
JUSTIN BIEBER FLOOR CLEANING.
```

Problem 3 Programming Sports for all (allidrett) (30%)

(In this problem, it can be useful to call functions you have made in earlier sub-problems, even if you have not solved them and assume that they work as specified in the problem description text).

Lea and Lars are coaches for sports for all for 2nd graders at Pythonmyra school. They participate in mini-tournaments in soccer and indoor bandy, with 3-5 players on the field and additional 1-2 substitutes. As it requires time to manually workout plans for assigning players to teams and frequency of substitute players, they want software that can assist them doing the job.

Problem 3a (4%)

It is desired that all the players get equal time to play in a match if you have substitutes on a team. Write the function `sek_paa_benken(ant_paa_laget,ant_paa_banen,kamptid)`. The input parameters are the number of players on the team, how many should be on the field in a match, and the length of a match given in number of minutes. The function shall return the number of seconds each player must be a substitute during a match, rounded to closest integer. Example of execution:

```
>>> sek_paa_benken(6, 5, 12)
120
>>> sek_paa_benken(6, 4, 12)
240
>>> sek_paa_benken(7, 5, 12)
206
>>>sek_paa_benken(5, 5, 12)
0
>>>
```

Problem 3b (4%)

When a coach look at his watch during matches to check when next substitute will happen, it is easier to relate to minutes and seconds instead of just seconds. Write the function `minutt_sekund(sekunder)` where the input parameter is the number of seconds `(sekunder)` (integer) and the return value is a string formatted as '`mm:ss`'. If the number of minutes is less than 10, the string should be formatted as '`m:ss`', but seconds should always be represented as to characters, also if it is less than 10. Example of execution.

```
>>> minutt_sekund(120)
'2:00'
>>> minutt_sekund(206)
'3:26'
>>>
```

Problem 3c (4%)
It happens that some players notify that they cannot play in a cup (no-shows). Write the function `les_inn_forfall( )` which lets the user write names using the keyboard, or an empty string ('') to quit (no more no-shows). The names the user input, shall be put in a list, which is returned from the function. You can assume that the user does not make any typing error and all input is correct. Example of execution (the leading text is printed by the function, the names are entered by the user, and the list is what the function returns):

```
>>> les_inn_forfall()
Skriv navn, eller kun ENTER (tom tekst) for å avslutte.
Spiller som har meldt forfall: Henrik
Spiller som har meldt forfall: Emma B.
Spiller som har meldt forfall: Lucas
Spiller som har meldt forfall:
['Henrik', 'Emma B.', 'Lucas']
>>>
```

Problem 3d (4%)
Write the function `finn_tilgjengelige(alle, forfall)` which has two lists input parameters, where the first is all the children signed up for sports for all, and the second is the list of those who cannot participate in a specified cup (no-shows). The function shall return a new list that contains the children available to play in the specific cup. Note: The function must not change the content of list of the input parameter `alle`.

Example of execution:

```
>>> barn = ['Ada', 'Bo', 'Emma A.', 'Emma B.', 'Henrik',  'Ine', 'Jo', 'Kim',
'Lucas', 'My', 'Ola', 'Pia']
>>> forfall = ['Henrik', 'Emma B.', 'Lucas']
>>> finn_tilgjengelige(barn, forfall)
['Ada', 'Bo', 'Emma A.', 'Ine', 'Jo', 'Kim', 'My', 'Ola', 'Pia']
>>>
```

Problem 3e (6%)
Some children (an partly parents) have strong opinions about who should be on the team together, and this is a source of conflict. To minimalize this problem, the coach want the software to automatically come up with a suggestion to how players are assigned to teams, based on random assignment of players. Write the function `laginndeling(spillere, sp_per_lag)` where the input parameter `spillere` is the list of available players for a specific cup, and `sp_per_lag` is the number of players allowed on the playing field. The function shall return a list of lists, where the outer list contain the teams who will play in this cup, and the inner lists contain names of the players on each of the teams. All the teams must as least have the number of players specified by the input parameter `sp_per_lag`, and no person can be assigned to more than one team. The function shall produce the maximum of teams that can play (meaning fewest number of substitutes), so the children get as much playing time as possible.

Example of execution: 14 children will participate in a cup where each team must have 4 players on the playing field. The function has randomly assigned the children on 3 teams, where 2 have 5 players (1 substitute) and one has 4 players (0 substitutes). The team layout is returned as a list of lists.

```
>>> sp = ['Ada', 'Bo', 'Eli', 'Isa', 'Cindy', 'Henrik',  'Ine', 'Jo', 'Kim',
'Lucas', 'My', 'Noor',  'Ola', 'Pia']
>>> laginndeling(sp, 4)
[['Eli', 'Henrik', 'Pia', 'Ada', 'Ine'], ['Kim', 'My', 'Ola', 'Lucas', 'Noor'],
['Isa', 'Bo', 'Cindy', 'Jo']]
>>>
```

Problem 3f (5%)
Write the function `main()` which from the user reads in which children who has notified that they cannot come, how many players there will be on the playing field (per team), and how long the matches are (minutes) for a given cup. You can assume the global variable `BARN`, which contains a list of all the children involved in Sports for all, so they do not need to be inputted by the user. Based on this given information, `main()` will print out to screen the information shown below (the example shows execution where `BARN` contains Jo and Henrik, as well as they who are listed in the teams below):

```
Skriv navn, eller kun ENTER (tom tekst) for å avslutte.
Spiller som har meldt forfall: Jo
Spiller som har meldt forfall: Henrik
Spiller som har meldt forfall:
Spillere per lag: 5
Kamptid (minutter): 15
Lag 1 :
['Pia', 'Bo', 'Ada', 'Lucas', 'Emma A.', 'Cindy']
Tid på benken per spiller: 2:30

Lag 2 :
['Emma B.', 'Yngve', 'Ola', 'My', 'Quentin', 'Sara']
Tid på benken per spiller: 2:30

Lag 3 :
['Noor', 'Kim', 'Tuva', 'Rashad', 'Ine']
Tid på benken per spiller: 0:00
```

Problem 3g (5%)
Each mini-cup is arranged so that each team plays a number of matches, typically 3-4. The organization of the matches for the cup is broadcasted by the arranger on a text file formatted in a such way that every line contains time (for the match to start), the two teams who will play against each other, and the name of the playing field for the match. The coaches want the software to have a function, `ny_fil()`, which reads the file how of the matches are organized (kampoppsett.txt), and writes a new text file on the same format – ourGames.txt – which ONLY contains the matches where teams from Pythonmyra is involved. Make the function with three input parameters: One for the name of the team (e.g. Pythonmyra), one for the name of the input file (e.g. kampoppsett.txt) and one for the name of the output file (e.g. ourGames.txt).

Example of excerpt from the file kampoppsett.txt:

```
17:20 Pythonmyra 1 - Ranheim 2 (Bane 3)
17:20 Fredig 1 - Ranheim 3 (Bane 1)
17:20 Ranheim 1 - Utleira 2 (Bane 2)
17:20 Astor 1 - Vestbyen (Bane 4)
17:40 Freidig 4 - Pythonmyra 2 (Bane 1)
17:40 Trond 1 – Freidig 2 (Bane 2)
17:40 Pythonmyra 3 – Trond 2 (Bane 3)
18:00 Utleira 2 – Pythonmyra 1 (Bane 2)
```

The file given above, the function will produce a new file as follows:

```
17:20 Pythonmyra 1 - Ranheim 2 (Bane 3)
17:40 Freidig 4 - Pythonmyra 2 (Bane 1)
17:40 Pythonmyra 3 – Trond 2 (Bane 3)
18:00 Utleira 2 – Pythonmyra 1 (Bane 2)
```

Problem 4 Understanding code (20%)

Problem 4a (5%)

What will be printed to screen if you call `a(5)` with the code below? (3 %)
Explain with one sentence what the function `a()` does? (2 %)

```
def a(b):
    for c in range(1,b+1):
        for d in range(1,b+1):
            if (d<b):
                print(c*d, end=',')
            else:
                print(c*d)
```

Problem 4b (5%)

What will be returned if you call `f([[3,5],[2,4],[1,3]])` with the code below? (3 %)
Explain with one sentence what the function `f()` does? (2 %)

```
def f(b):
    c=len(b[0])
    d=len(b)
    g = [[0 for row in range(d)]
       for col in range(c)]
    for e in range(0,c):
       for f in range(0,d):
          g[e][f]=b[f][e]
    return g
```

Problem 4c (5%)

What will be returned if you call `u(5)` with the code below? (3 %)
Explain with one sentence what the function `u()` does? (2 %)

```
def u(x):
  if x <=1:
    return 1
  else:
    return x*u(x-1)
```

Problem 4d (5%)
What will be returned if you call `nrk('Nylnpuokrtrjhtrsklkiok')` with the code below? (3 %)
Explain with one sentence what the function `nrk()` does? (2 %)

```
def nrk(tekst):
  s=''
  i=0
  j=1
  while i<len(tekst):
    s+=tekst[i]
    i=i+j
    j=j+1
  return s
```

Appendix: Useful Functions and Methods

Built-in:

format(numeric_value, format_specifier)
: Formats a numeric value into a string according to the format specifier, which is a string that contains special characters specifying how the numeric value should be formatted. Examples of various formatting characters are "f=floating-point, e=scientific notation, %=percentage, d=integer". A number before the formatting character will specify the field width. A number after the character "." will format the number of decimals.

%
: Remainder (modulo operator): Divides one number by another and gives the remainder.

len(s)
: Return the length (the number of items) of a string, tuple, list, dictionary or other data structure.

int(x)
: Convert a string or number to a plain integer.

float(x)
: Convert a string or a number to floating point number.

str([object])
: Return a string containing a nicely printable representation of an object.

String methods:

s.isalnum()
: Returns true if the string contains only alphabetic letters or digits and is at least one character of length. Returns false otherwise.

s.isalpha()
: Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.

s.isdigit()
: Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.

s.isspace()
: Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (\n), and tabs (\t)).

s.ljust(width)
: Return the string left justified in a string of length width.

s.rjust(width)
: Return the string right justified in a string of length width.

s.lower()
: Returns a copy of the string with all alphabetic letters converted to lowercase.

s.upper()
: Returns a copy of the string with all alphabetic letters converted to uppercase.

s.strip()
: Returns a copy of the string with all leading and trailing white space characters removed.

s.strip(char)
: Returns a copy of the string with all instances of *char* that appear at the beginning and the end of the string removed.

s.split(str)
: Returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified).

s.endswith(substring)
: The substring argument is a string. The method returns true if the string ends with substring.

s.startswith(substring)
: The substring argument is a string. The method returns true if the string starts with substring.

s.find(substring)
: The substring argument is a string. The method returns the lowest index in the string where substring is found. If substring is not found the method returns -1.

s.replace(old, new)
: The old and new arguments are both strings. The method returns a copy of the string with all instances of old replaced by new.

List operations:

s[i:j:k]
    Return slice starting at position i extending to position j in k steps. Can also be used for strings.

item in s
    Determine whether a specified item is contained in a list.

min(list)
    Returns the item that has the lowest value in the sequence.

max(list)
    Returns the item that has the highest value in the sequence.

s.append(x)
    Append new element x to end of s.

s.insert(index,item)
    Insert an item into a list at a specified position given by an index.

s.index(*item*)
    Return the index of the first element in the list containing the specified item.

s.pop()
    Return last element and remove it from the list.

s.pop(i)
    Return element i and remove it from the list.

s.remove(item)
    Removes the first element containing the item.

s.reverse()
    Reverses the order of the items in a list.

s.sort()
    Rearranges the elements of a list so they appear in ascending order.

Dictionary operations:

d.clear()
    Clears the contents of a dictionary

d.get(key, default)
    Gets the value associated with a specific key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.

d.items()
    Returns all the keys in a dictionary and their associated values as a sequence of tuples.

d.keys()
    Returns all the keys in a dictionary as a sequence of tuples.

d.pop(key, default)
    Returns the value associated with a specific key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.

d.popitem()
    Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.

d.values()
    Returns all the values in dictionary as a sequence of tuples.

Files

open()
    Returns a file object, and is most commonly used with two arguments: open(filename, mode). Mode can be ‘r’ (read only), ‘w’ (writing only), ‘a’ (appending), ‘r+’ (both reading and writing).

f.read(size)
    Reads data from file and returns it as a string. Size is an optional and if left out the whole file will be read.

f.readline()
    Reads a single line from the file (reads until newline character (\n) is found), and returns it as a string.

f.readlines()
    Reads data from the file and returns it as a list of strings.

f.write(string)
    Writes the contents of string to file.

f.close()
    Close the file and free up any system resources taken up by the open file.

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				

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				