Module Title:	Programming for Data Analysis
Module Code:	B9DA108
Module Leader:	Clive Gargan & Paul Laird
Assessment Title:	Algorithm Design & Programming
Assessment Number (if relevant):	1
Assessment Type:	Practical (lab-based)
Restrictions on Time/Length:	N/A
Individual/Group:	Individual
Assessment Weighting:	30%
Issue Date:	Week ending 11th March
Hand In Date:	25th March (6pm)
Planned Feedback/Results Release Date:	Within three weeks of submission
Mode of Submission:	On-line **ONLY Moodle**

The assignment is an individual assignment.

Requirements: You are required to submit Python programs for each of the four problems given in this document. For each Python program, you are required to use appropriate variable names (follow the naming conventions). Also make use of comments in your code.

Submission: Zip the Solution and Project folder and upload using the CA_ONE link on Moodle before 6pm on 25th March.

IMPORTANT NOTE: Only one submission is allowed so please ensure that you have everything that is required in the zip before uploading to Moodle.

The following table illustrates the percentage allocation for each individual part of the assignment.

<u>Total Marks (100)</u>

Problem	**Detail**	**Breakdown of Marks**
Problem 1	Python Code	35 marks
Problem 2	Python Code	15 marks
Problem 3	Python Code	25 marks
Problem 4	Python Code	25 marks

IMPORTANT NOTE: All of your Python programs should be compiling at a minimum and you must target Python version 3.

Problem 1

Write a program in Python to prompt the user to input their name, employee number, week ending date, hours worked, rate per hour, standard and overtime tax percentage rate. Use the data input to calculate gross pay, tax deductions and net pay. Output the results as a formatted payslip. Assume that a standard working week is 37.5 hours.

E.g. Ask the user to enter the following data:

Employee Name:	(sample input – Mark Bate)
Employee Number:	(sample input – 123456789A)
Week ending:	(sample input - 26/01/2018)
Number of hours worked:	(sample input – 42.5)
Hourly Rate:	(sample input – 10.50)
Overtime Rate:	(time-and-a-half as 1.5)
Standard Tax Rate:	(sample input – 20)
Overtime Tax Rate:	(sample input – 50)

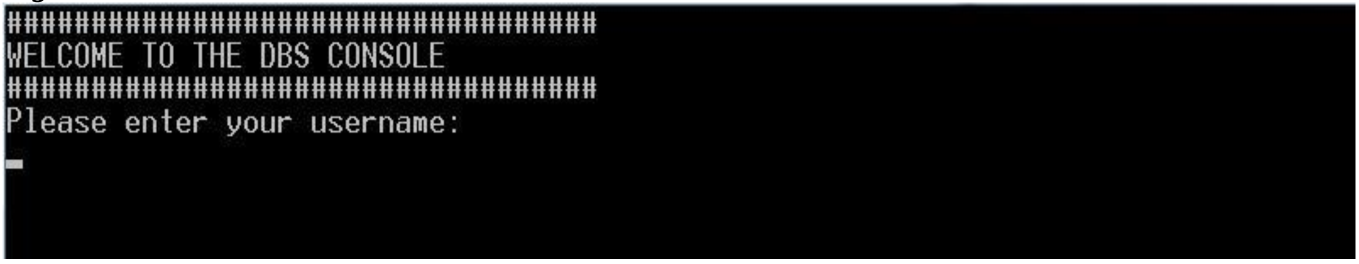
Once the above data has been entered the program should display the employee's payslip as per the following example:

```
                             PAYSLIP
WEEK ENDING 26/01/2018
Employee: Mark Bate
Employee Number: 123456789A
                      Earnings                Deductions
                      Hours  Rate    Total
Hours (normal)        37.50  10.50   393.75  Tax @ 20%  78.75
Hours (overtime)      5.00   15.75    78.75  Tax @ 50%  39.37

                      Total pay:                       472.50
                      Total deductions:                118.12
                      Net pay:                         354.38
```

Problem 2

Write a program in Python which prompts the user for their username in the format *Domain Name\Username* as per Figure 1a below.

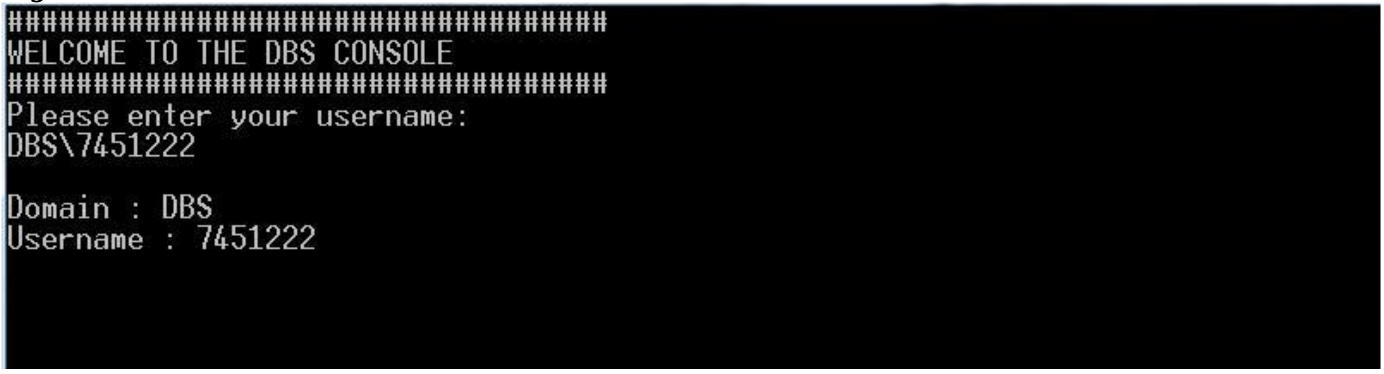
Figure 1a.



```
######################################
WELCOME TO THE DBS CONSOLE
######################################
Please enter your username:
_
```

On entering their domain and username and pressing carriage return, write out to the console window each individual data item as per Figure 1b below.

Figure 1b.



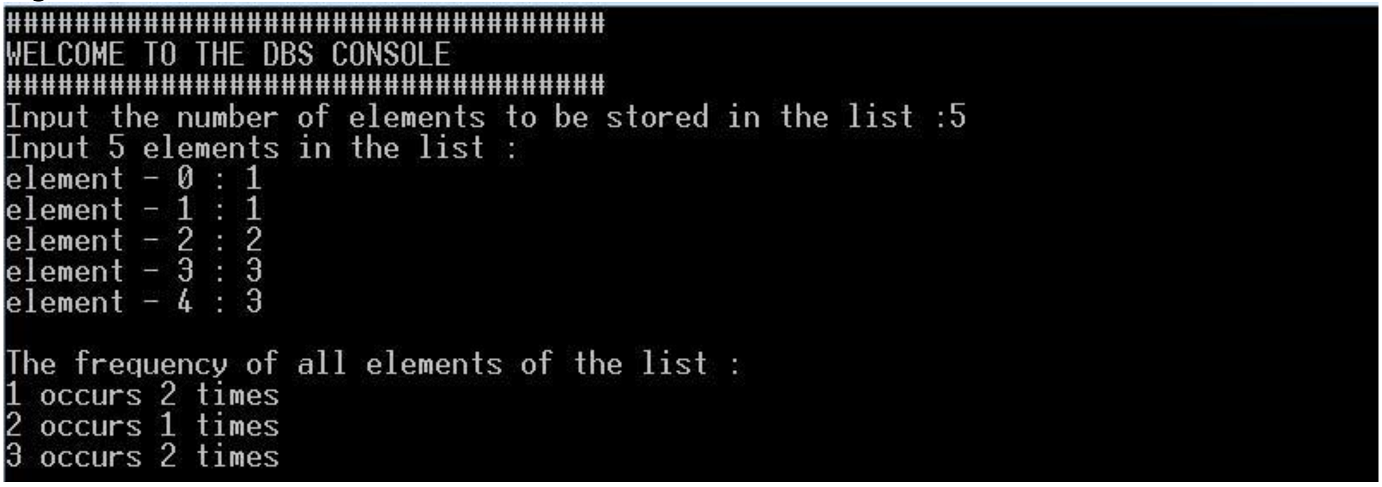
```
######################################
WELCOME TO THE DBS CONSOLE
######################################
Please enter your username:
DBS\7451222

Domain : DBS
Username : 7451222
```

NOTE: The user can enter any combination of domain and username.

Problem 3

Write a program in Python that prompts the user to enter a number of integer values. The program stores the integers, counts the frequency of each integer and displays the frequency as per Figure 2 below.

Figure 2

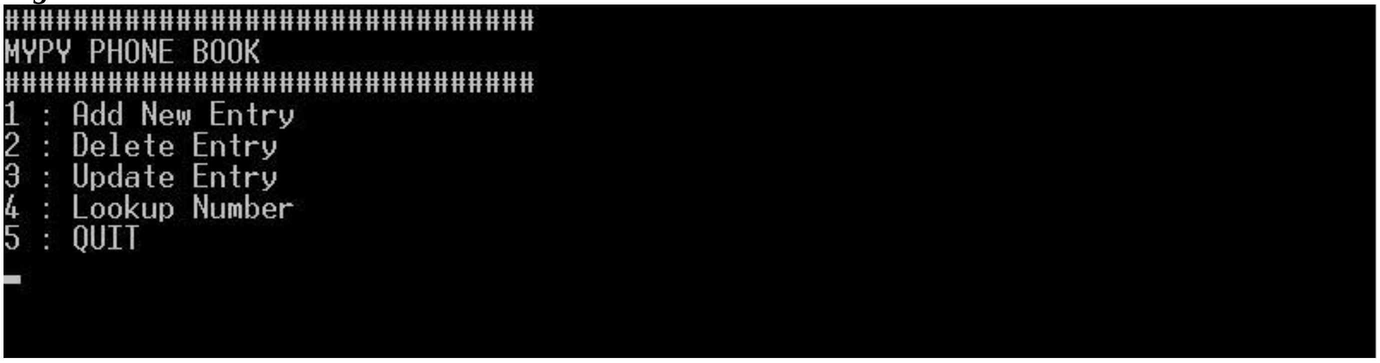


```
####################################
WELCOME TO THE DBS CONSOLE
####################################
Input the number of elements to be stored in the list :5
Input 5 elements in the list :
element - 0 : 1
element - 1 : 1
element - 2 : 2
element - 3 : 3
element - 4 : 3

The frequency of all elements of the list :
1 occurs 2 times
2 occurs 1 times
3 occurs 2 times
```

Problem 4

Implement the MYPY Phone Book System in Python as per Figure 3 below which allows users to add, delete, update and lookup phone numbers. The MYPY Phone Book System should store the individual's Full Name and Phone Number. Your program should not allow users to add the same number twice. On adding, deleting, updating or looking up a number, your program should let the user know if the operation was successful or not. On looking up a number return the full name and number of the individual; if the number is not found give the user the option to add the details they are looking up. The user can perform multiple actions; they can add a new entry and subsequently delete an entry without having to stop and start the program until they decide to quit.

Figure 3



```
##################################
MYPY PHONE BOOK
##################################
1 : Add New Entry
2 : Delete Entry
3 : Update Entry
4 : Lookup Number
5 : QUIT
```

Penalty for Late Submissions

The number of marks deducted depends on the lateness of the submission and will be deducted according to the following scale:

- Where an assessment is submitted between 1 and 14 days late 2 marks per day are deducted
- An assessment submitted after the deadline but within 24 hours of the original deadline will attract the first day penalty, i.e. deduction of 2 marks
- Where an assessment is more than 14 days late it is annotated at the discretion of the lecturer but no marks are awarded.

Assessment criteria

Criteria/ Mark	**< 40**	**40-49**	**50-59**	**60–69**	**70 +**
Python code	Insufficient or incomplete code structure	Some but insufficient and poorly structured code which doesn't solve the problem	Sufficient solves problem but lack of attention to coding practices	Well-structured code	Excellent solution to problem