|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **logobw**  CANDIDATE NAME  CT GROUP | VICTORIA JUNIOR COLLEGE  JC 2 COMMON TEST  Higher 2  **………………………………………….………………………………….…………..**  **……………………………..** | | | |
| **COMPUTING** | | | **9569/02** | |
| **Paper 2**  Additional Materials: .txt data file | | | | **17 May 2023**  **2 hours** |
|  | |  | | |
| Additional Materials: Electronic version of EMPLOYEE.txt data file  Electronic version of DEPARTMENT.txt data file  Electronic version of EMPLOYEE\_DEPLOYMENT.txt data file  Insert Quick Reference Guide  **READ THESE INSTRUCTIONS FIRST**  Answer **all** questions.  All tasks must be done in the computer laboratory.  Approved calculators are allowed. The use of built-in functions, where appropriate, is allowed for this paper unless stated otherwise.  Save each task as it is completed.  Note that up to **6** marks out of 65 will be awarded for the use of common coding standards for programming style.  The numbers of marks is given in brackets [ ] at the end of each question or part question. The total number of marks for this paper is 65.   |  |  | | --- | --- | | For Examiner’s Use | | | Total | **/ 65** | | | | | |

|  |  |
| --- | --- |
|  | |
|  | This document consists of **7** printed pages and **0** blank page. |  |

**Instruction to candidates:**

Your program code and output for each of Task 1 to 3 should be saved in a single .ipynb file using Jupyter Notebook. For example, your program code and output for Task 1 should be saved as:

TASK1\_<your name>\_<class>\_<index number>.ipynb

Make sure that each of your .ipynb files shows the required output in Jupyter Notebook.

1. Name your Jupyter Notebook as:

TASK1\_<your name>\_<class>\_<index number>.ipynb

The task is to implement a ROT13 cipher encryption algorithm.

A ROT13 (“rotate by 13 places”) cipher is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the English alphabet.

For example:

* The letter 'A' is replaced with 'N' and so on up to ‘M’ which is replaced with ‘Z’.
* For letters beyond ‘M’, the sequence continues at the beginning of the alphabet. For example, the letter 'N' is replaced with 'A'.
* The same is done for lowercase letters. The letter ‘a’ is replaced with ‘n’ and the letter ‘z’ is replaced with ‘m’.
* Numbers, symbols, punctuation, space and all other characters are left unchanged.

Since there are 26 letters in the English alphabet, ROT13 cipher is a reciprocal cipher. This means that applying the ROT13 algorithm to the encrypted text will decrypt it.

The following table shows the ASCII values of some of the English alphabet.

|  |  |
| --- | --- |
| **Character** | **ASCII value** |
| A | 65 |
| Z | 90 |
| a | 97 |
| z | 122 |

In ASCII, the letters follow on numerically. For example, the letter ‘A’ is 65, 'B' is 66, 'C' is 67 etc.

For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

|  |  |
| --- | --- |
| ln [1]: | *#Task 1.1*  *Program code* |
|  | Output: |

**Task 1.1**

This program will encrypt each letter by replacing the letter with the 13th letter after it in the English alphabet. If the character is not a letter, the same character is returned.

Write program code for a function that takes a character as a parameter and returns the ASCII value of the new encrypted character. [5]

**Task 1.2**

Write program code to:

* read in a single character from the user
* call your function from **Task 1.1** with this character
* output its encrypted character or the same character if the character is not a letter. [3]

Test your program **four** times, with the following test data. Show all **four** outputs.

A

a

3

# [2]

**Task 1.3**

Write program code to:

* read in a text message from the user
* use your function from **Task 1.1** to encrypt each character
* output the encrypted message. [4]

Test your program with the following test data:

Why did the chicken cross 20 Marine Vista? [1]

Save your Jupyter Notebook for Task 1.

1. Name your Jupyter Notebook as:

TASK2\_<your name>\_<class>\_<index number>.ipynb

This task is to implement a binary search tree Abstract Data Type (ADT) using Object-Oriented Programming (OOP). The tree is used to store integer values in descending numerical order.

The program to implement this ADT will use the class Tree designed as follows:

The class Tree contains the following methods:

* a constructor
* an insert method to take the parameter and store it in the correct position in the tree
* a recursive print method that uses pre-order traversal to output the following in the tree
  + the data item
  + the left pointer data item
  + the right pointer data item

The class Tree contains three properties:

* data is the integer value
* left\_pointer points to the left subtree
* right\_pointer points to the right subtree

For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

|  |  |
| --- | --- |
| ln [1]: | *#Task 2.1*  *Program code* |
|  | Output: |

**Task 2.1**

Write program code to declare the class Tree and the constructor. [3]

Write the method to insert a new node into the tree for the Tree class. [4]

Write the method to output the data, left pointer’s data and right pointer’s data of each node for the Tree class. [3]

**Task 2.2**

Write a function quicksort(lst) that sorts integer values in list lst in descending order using quicksort algorithm. It returns the ordered integers in a list. [3]

Test your program by using the following test data:

[25, 70, 20, 90, 55, 45, 65, 35, 75, 40, 30, 60, 10, 80] [1]

**Task 2.3**

Write the main program to:

* declare a new instance of Tree
* call the function quicksort(lst) using the test data in **Task 2.2** and store them in a list
* store each unique value of the list as a new node in the tree. [3]

Test your program by showing the output of the print method of the tree. [1]

1. Name your Jupyter Notebook as:

TASK3\_<your name>\_<class>\_<index number>.ipynb

A new startup company wants to create a suitable database to store records about its employees, departments and employee’s deployment. The database should allow the company to run searches for specific data.

The database will have three tables: Employee, Department and Employee\_deployment.

The fields in each table are:

Employee:

* EmployeeID – unique employee number, for example, E101
* Employee\_name – the name of the employee
* Home\_address – the home address of the employee

Department:

* DepartmentID – unique department number, for example, D201
* Department\_name – the name of the department
* Location – location of the department
* Manager – manager of the department

Employee\_deployment:

* EmployeeID – the employee’s unique number
* DepartmentID – the department’s unique number
* Date\_of\_joining – the date that the employee joined the department
* Active – TRUE if employee is still serving in that department, or FALSE if employee is no longer in that department

A unique entry in the Employee\_deployment table will consist of the EmployeeID, DepartmentID and Date\_of\_joining fields.

For each of the sub-tasks 3.1 to 3.3, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

|  |  |
| --- | --- |
| ln [1]: | *#Task 3.1*  *Program code* |
|  | Output: |

**Task 3.1**

Write a Python program that uses SQL code to create the database COMPANY with the three

tables given. Define the primary and foreign keys for each table. [6]

**Task 3.2**

The text files EMPLOYEE.txt, DEPARTMENT.txt and EMPLOYEE\_DEPLOYMENT.txt store the comma- separated values for each of the tables in the database.

Write a Python program to read in the data from each file and then store each item of data in the correct place in the database. [5]

**Task 3.3**

Write a Python program to input an employee's number and return all the departments that the employee has been deployed in and if employee is still serving in that department. [5]

Test your program by running the application with the employee number 'E108'. [2]

Save your Jupyter Notebook.

**Task 3.4**

The company wants to filter the employees by Department and display the results in a web browser.

Write a Python program and the necessary files to create a web application that:

* receives a Department string from a HTML form, then
* returns a HTML document that enables the web browser to display a table tabulating the details of the employees currently deployed in that Department, in chronological order of Date\_of\_joining.

The table will display the following data:

* Employee’s name
* Date of joining the department

Save your Python program as:

TASK3\_4\_<your name>\_<class>\_<index number>.py

with any additional files/ subfolders in a folder named:

TASK3\_4\_<your name>\_<class>\_<index number> [6]

Run the web application. Save the webpage output of the program when 'Operations' is entered as the Department as:

TASK3\_4\_<your name>\_<class>\_<index number>.html [2]

**End of Paper**