|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **logobw**  CANDIDATE NAME  CT GROUP | VICTORIA JUNIOR COLLEGE  JC 2 TIMED PRACTICE  Higher 2  **……………………………………………….…………..**  **……………………………..** | | | |
| **COMPUTING** | | | **9569/02** | |
| **Paper 2**  Additional Materials:  Electronic version of MAZE.txt data file  Electronic version of LAPTOPS.txt data file  Electronic version of MONITORS.txt data file  Electronic version of PRINTERS.txt data file  Insert Quick Reference Guide | | | | **8 March 2023**  **1 hour 30 minutes** |
|  | |  | | |
| **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 **4** marks out of 54 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 54.   |  |  | | --- | --- | | For Examiner’s Use | | | Total | **/ 54** | | | | | |

|  |  |
| --- | --- |
|  | |
|  | This document consists of **6** printed pages and **0** blank page. |  |

1 In a computer game, a player (ʺOʺ) moves around a maze measuring 10 metres by 11 metres to collect a prize (ʺPʺ). The prize is placed at a random position within the maze. The prize position is not where a wall (ʺXʺ) appears in the maze. An empty position is indicated with a full-stop (ʺ.ʺ).

The maze is represented on the screen by a rectangle grid. Each square metre of the maze is represented by an x-coordinate and a y-coordinate. The top left square metre of the puzzle display has x = 0 and y = 0.

The player moves left, right, up or down according to a direction entered by the user. The game is turn-based; a user enters the direction, the player moves one position in that direction. If the direction would place the player on a wall, then the player does not move. The maze is displayed after each move.

XXXXXXXXXX

X........X

X.X.X.XX.X

X.X..P...X

X.XXXXXX.X

X...O....X

X.X.XX.X.X

X.X....X.X

X.XX.XXX.X

X........X

XXXXXXXXXX

To move the character, the user is prompted for a direction. The following are valid inputs:

|  |  |
| --- | --- |
| **Input character** | **Action** |
| ʺUʺ | Player moves up |
| ʺDʺ | Player moves down |
| ʺLʺ | Player moves left |
| ʺRʺ | Player moves right |
| ʺʺ | Continue with previous move.  If no previous move, do nothing |

When a player enters the move, a new position for the player is calculated and the maze is displayed. The previous position is changed back to a ʺ.ʺ when the player has a new position.

The moves are repeated until the player is at the same position as the prize. When the player and the prize are at the same position, the message “Player has reached the prize” is displayed and the game ends.

The maze is implemented using Object-Oriented Programming (OOP).

The class Maze contains three properties:

* maze – a 2-dimension list that contains the grid of the maze
* player – a dictionary that contains the coordinates of the player in the maze
* previous – a string that keeps track of previous move

The class Maze contains the following methods:

* a constructor to initialize maze with data from MAZE.TXT, place the prize randomly on the maze and place the player in the starting position x = 4, y = 5 on the maze.
* a move method that
  + takes in a string value as its parameter to indicate direction of move of player
  + calculate a new position of player
  + check that new position is not a wall
  + update the new and previous position of player on the grid correctly
  + returns True if player and the prize are at the same position, else returns False.
* a display\_maze method that displays the maze with the position of the player and prize.

**Task 1.1**

Write program code in Python to define the class Maze. [12]

The maze is given in the text file MAZE.TXT. You may read in the data from this file or place the data in your program using any suitable method.

**Task 1.2**

Write the main program to:

* declare a new instance of Maze
* prompt player for a direction and validate the direction
* use the move method to move player if the direction entered is valid
* display the maze
* continue until the player is at the same position as the prize
* print the message “Player has reached the prize” when the player is at the same position as the prize and the game ends. [10]

**Task 1.3**

A new feature of counting the number of steps a player takes to reach the prize is to be added into the game. Each valid move the player takes is counted as one step.

The subclass Maze2 is used to implement this feature. An additional property count is added to the subclass.

The subclass Maze2 contains the following methods:

* get\_count() method returns the number of steps taken by player
* move() method does the same as the method in the parent class and in addition, counts the number of steps a player takes to reach the prize.

Write program code for the subclass Maze2.

Use appropriate inheritance and polymorphism in your designs. [4]

2 A large company currently keeps records on paper of all the computing equipment it owns. Every computer device has its information recorded when it is purchased.

The company decided to trial a database to manage its computing equipment records. It is expected that the database should be normalised.

When a computer device is purchased, the following information is recorded:

* serial\_number – unique serial number of device
* type – type of device ('Monitor', 'Laptop', or 'Printer')
* model – model of device
* location – where the device is used
* date\_of\_purchase – date of purchase
* written\_off – whether the device is still in use (‘True’ means device is written off and NOT in use, ‘False’ means device is still in use)

For monitors, the following extra information is recorded:

* date\_cleaned – the last date the monitor was cleaned

For laptops, the following extra information is recorded:

* weight\_kg – the weight in kilograms

For printers, the following extra information is recorded:

* toner – type of toner required
* date\_changed – the last date the toner cartridge was changed

The information is to be stored in four different tables:

Device

Monitor

Laptop

Printer

**Task 2.1**

Create an SQL file called TASK2\_1\_<your name>\_<CG>\_<index number>.sql to show the SQL code to create the database equipment.db with the four tables. The table, Device, must use serial\_number as its **primary key**. The other tables must refer to the serial\_number as a **foreign key**.

Save your SQL code as

TASK2\_1\_<your name>\_<CG>\_<index number>.sql [5]

**Task 2.2**

The files MONITORS.txt, LAPTOPS.txt and PRINTERS.txt contain information about the company’s monitors, laptops, and printers respectively for insertion into the equipment database. Each row in the three files is a comma-separated list of information about a single device.

For MONITORS.txt, information about each monitor is given in the following order:

serial\_number, model, location, date\_of\_purchase, written\_off, date\_cleaned

For LAPTOPS.txt, information about each laptop is given in the following order:

serial\_number, model, location, date\_of\_purchase, written\_off, weight\_kg

For PRINTERS.txt, information about each printer is given in the following order:

serial\_number, model, location, date\_of\_purchase, written\_off, toner, date\_changed

Write a Python program to insert all information from the three files into the equipment database, equipment.db. Run the program.

Save your program code as

TASK2\_2\_<your name>\_<CG>\_<index number>.py [5]

**Task 2.3**

Write SQL code to show the serial number, model and the location of each monitor, with the date it was last cleaned. Run this query.

Save this code as

TASK2\_3\_<your name>\_<CG>\_<index number>.sql [4]

**Task 2.4**

The company wants to filter the devices by Location and the display results in a web browser.

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

* receives a Location string from a HTML form, then
* creates and returns a HTML document that enables the web browser to display a table tabulating the serial\_number and Type of devices still in use at that exact Location.

Save your python program as:

Task\_2\_4\_<your name>\_<CG>\_<index number>.py

With any additional files / sub\_folders as needed in a folder named

Task\_2\_4\_<your name>\_<CG>\_<index number>

Run the web application.

Save the webpate output of the program when “Office 51” is entered as the Location as TASK2\_4\_<your name>\_<CG>\_<index\_number>.html [10]