RESTRICTED

H

MINISTRY OF EDUCATION, SINGAPORE
in collaboration with
UNIVERSITY OF CAMBRIDGE LOCAL EXAMINATIONS SYNDICATE
General Certificate of Education Advanced Level
Higher 2

COMPUTING

9569/02

Paper 2

For Examination from 2020

SPECIMEN MARKING GUIDE FOR TEACHERS

3 hours

MAXIMUM MARK: 100

This document consists of **11** printed pages and **1** blank page.

Singapore Examinations and Assessment Board

CAMBRIDGE
International Examinations

[Turn over

Where responses in addition to those given in the marking guide are possible, full marks will be given for a correct response, with equivalent sub-marks for equivalent stages. (This does not however apply if candidates are directed in the question to answer a question in a particular way.)

Task	Answer	Marks
1.1	Prompt and input [1] Test in ranges A–Z and a–z and suitable error message out of range… [1] Allow reinput [1] Display correct character only [1] ***Test program*** Display `A`, `b` [1] Display error message for = [1]	**6**
1.2	Conversion to denary (ord) [1] Input of allowable number base only [1] Conversion to number base input (use of div and mod) [1] Use of `A`, `B`, `C` and `D` for appropriate base (`A` for `11`, `A` and `B` for `12`, `A`, `B` and `C` for `13`, `A`, `B`, `C` and `D` for `14`) [1] Screen displays match sample [1] ***Test program*** For `B` and `11`, correct values `B`, `66`, `60` [1] For `m` and `14`, correct values `m`, `109`, `7B` [1]	**7**
1.3	Accurate menu display [1] Acceptance of correct menu choices only [1] Activation of appropriate code for four correct menu choices [1] Activation of appropriate code for all correct menu choices [1] ***Test program*** Menu choice `1`, followed by `X`, menu choice `5`, output `6A`, Menu choice `3`, output `80` [1] Menu choice `1`, followed by `y`, menu choice `2`, output `121` [1] Menu choice `7` program closes [1]	**7**

Task	Answer	Marks
2.1	Read file into suitable array [1] Use of quick sort [1] Pick a pivot [1] Reorder so that all elements less than the pivot are before the pivot [1] All elements greater than the pivot are after the pivot (= can go either way) [1] Repeat for each sub- array above and below the pivot [1] Until each sub-array is of length 1 or 0 [1] Store new list in file [1] Output of the program showing list sorted in ascending order of file name [1] `TESTDATA_Filename_order.txt` contains list in correct order [1]	**10**
2.2	At least one appropriate prompt for input … [1] an output with an appropriate message … [1] … all inputs have appropriate prompts [1] … all outputs have appropriate messages [1] Some errors are tested and appropriate messages are output … [1] … all errors are tested and appropriate messages are output [1] At least one option is written as a separate procedure [1] All options are written as separate procedures [1] ***A different sort must be chosen*** Sort attempts to display list in chronological date order correctly [1] Sort performs correctly … [1] … matches sort identified in comments [1] List searched for matching name [1] List searched for files with the same date modified [1] ***Test program*** File names sorted by chronological order of date displayed [1] Suitable test data chosen for the following tests to show: • File name found and displayed [1] • File name not found displayed [1] • Files update on a given date displayed [1] • No files updated on a given date displayed [1]	**18**

Task	Answer	Marks

3.1	• Superclass declaration [1]
	• Constructor sets tail, head and a list [1]
	• Display method outputs all elements in the list [1]
	
	e.g.
	`class Node:`
	`    def __init__(self, data, prev, next):`
	`        self.data = data`
	`        self.prev = prev`
	`        self.next = next`
	`class DataStructure:`
	`    def __init__(self):`
	`        self.head = None`
	`        self.tail = None`
	`    def is_empty(self):`
	`        return self.head is None`
	`    def insert(self, value):`
	`        if self.tail is None:`
	`            self.tail = Node(value, None, None)`
	`            self.head = self.tail`
	`        else:`
	`            self.tail.next = Node(value, self.tail, None)`
	`            self.tail = self.tail.next`
	
	`    def delete(self):`
	`        print("Not implemented")`
	
	`    def display(self):`
	`        if self.is_empty():`
	`            print("Empty structure")`
	`        else:`
	`            print_data = ""`
	`            current = self.head`
	`            while current is not None:`
	`                print_data += str(current.data) + ' '`
	`                current = current.next`
	`            print(print_data)`
		3

Task	Answer	Marks
3.2	• Subclass `Stack` and constructor inherits from superclass [1] • Insert method that adds item to stack, increments tail/head [1] • delete method that returns the value and decrements tail/head (depending which the candidate uses) [1] • Or appropriate value if empty [1] • New display method that overrides super method and outputs all elements in reverse order [1]	**5**

e.g.

```
class Stack(DataStructure): # inheritance
    def insert(self, value):
        if self.tail is None:
            self.tail = Node(value, None, None)
            self.head = self.tail
        else:
            self.tail.next = Node(value, self.tail, None)
            self.tail = self.tail.next
    def delete(self):
        if self.is_empty(): # inherited method
            return "Cannot delete from empty stack"
        else:
            return_value = self.tail.data
            self.tail = self.tail.prev
            if self.tail is None:
                self.head = None
            else:
                self.tail.next = None
            return return_value

    def display(self): # polymorphism
        if self.is_empty():
            print("Empty stack")
        else:
            print("Stack contents:")
            print_data = ""
            current = self.tail
            while current is not None:
                print_data += str(current.data) + ' '
                current = current.prev
            print(print_data)
```

Task	Answer	Marks

3.3	• Subclass queue and constructor inherits from superclass [1]
	• delete method returns the value [1]
	• … increments head (or removes item from list and decrements tail) [1]
	• … or appropriate value if empty [1]
	• insert method that adds item to queue, increments tail [1]
	
	e.g.
	`class Queue(DataStructure): # inheritance`
	
	`    def delete(self):`
	`        if self.is_empty():`
	`            return "Cannot delete from empty queue"`
	`        else:`
	`            return_value = self.head.data`
	`            self.head = self.head.next`
	`            if  self.head is None:`
	`                self.tail = None`
	`            else:`
	`                self.head.prev = None`
	`            return return_value`
	
	`    def insert(self, value):`
	`        if self.tail is None:`
	`            self.tail = Node(value, None, None)`
	`            self.head = self.tail`
	`        else:`
	`            self.tail.next = Node(value, self.tail, None)`
	`            self.tail = self.tail.next`
	
	`    def display(self): # polymorphism`
	`        if self.is_empty():`
	`            print("Empty queue")`
	`        else:`
	`            print("Queue contents:")`
	`            print_data = ""`
	`            current = self.head`
	`            while current is not None:`
	`                print_data += str(current.data) + ' '`
	`                current = current.next`
	`            print(print_data)`
		5

Task	Answer	Marks
3.4	• Stack created as object [1] • Queue created as object [1] • Opens file 'TASK3stack.txt' [1] • Reads all data into stack using appropriate method [1] • Opens file 'TASK3queue.txt' and reads all data into queue using appropriate method [1] • Display from superclass used to output both the stack and queue [1] • 2 items removed and output from stack [1] • 2 items removed and output from queue [1] • New contents of stack and queue output [1]	9

e.g.

```
# main
files = ["TASK3stack.txt", "TASK3queue.txt"]
# add stack and queue objects to generic data structure
list
data_structures = [Stack(), Queue()]
# insert file contents to stack and queue using
polymorphic insert method
for i in range(len(files)):
    file = open(files[i], 'r')
    lines = file.readlines()
    for line in lines:
        data_structures[i].insert(line.strip())
    file.close()
# display stack and queue contents using polymorphic
display method
for data_structure in data_structures:
    data_structure.display()
    # remove and output two items from stack and queue
using polymorphic delete method
    for i in range(2):
        print("Deleted:", data_structure.delete())
    data_structure.display()
    print()
```

Task	Answer	Marks

4.1	*Mark as follows:*
	Creation of 4 tables [1]
	Primary key of `SerialNumber` in `Device` [1]
	Foreign key of `SerialNumber` in other three tables [1]
	Correct columns in `Device` [1]
	Correct columns in other three tables [1]
	
	TASK4_1 sample SQL code for the database with four tables – seen either as SQL code or in a SQLite database file.
	
	`CREATE TABLE Device(`
	`    SerialNumber INTEGER NOT NULL PRIMARY KEY,`
	`    Type VARCHAR(20),`
	`    Model VARCHAR(20),`
	`    Location VARCHAR(20),`
	`    DateOfPurchase VARCHAR(20),`
	`    WrittenOff INTEGER`
	`);`
	
	`CREATE TABLE Laptop(`
	`    SerialNumber INTEGER NOT NULL,`
	`    WeightKg REAL,`
	`    FOREIGN KEY (SerialNumber)`
	`                REFERENCES Device(SerialNumber)`
	`);`
	
	`CREATE TABLE Monitor(`
	`    SerialNumber INTEGER NOT NULL,`
	`    DateCleaned VARCHAR(20),`
	`    FOREIGN KEY (SerialNumber)`
	`                REFERENCES Device(SerialNumber)`
	`);`
	
	`CREATE TABLE Printer(`
	`    SerialNumber INTEGER NOT NULL,`
	`    Toner VARCHAR(20),`
	`    DateChanged VARCHAR(20),`
	`    FOREIGN KEY (SerialNumber)`
	`                REFERENCES Device(SerialNumber)`
	`);`
		5

Task	Answer	Marks
4.2	Any 5 marks from: Create programmatic connection to database [1] Correct insertion of data into Device table [1] Correct data present from `MONITORS.txt` in `Monitor` table [1] Data present from `LAPTOPS.txt` in `Laptop` table [1] Data present from `PRINTERS.txt` in `Printer` table [1] Database transaction must be committed [1]	5

TASK4_2 possible sample Python code for insertion

```
import csv
import sqlite3

db = sqlite3.connect('equipment.db')

with open('MONITORS.txt') as monitors_file:
    monitors = csv.reader(monitors_file)
    for monitor in monitors:
        db.execute("INSERT INTO Device(SerialNumber, " +
            "Type, Model, Location, DateOfPurchase, " +
            "WrittenOff) VALUES(?, 'Monitor', ?, ?, " +
            "?, ?)", (monitor[0], monitor[1],
            monitor[2], monitor[3], monitor[4] ==
            'TRUE'))
        db.execute("INSERT INTO Monitor(SerialNumber, " +
            "DateCleaned) VALUES(?, ?)", (monitor[0],
            monitor[5]))

with open('LAPTOPS.txt') as laptops_file:
    laptops = csv.reader(laptops_file)
    for laptop in laptops:
        db.execute("INSERT INTO Device(SerialNumber, " +
            "Type, Model, Location, DateOfPurchase, " +
            "WrittenOff) VALUES(?, 'Laptop', ?, ?, " +
            "?, ?)", (laptop[0], laptop[1], laptop[2],
            laptop[3], laptop[4] == 'TRUE'))
        db.execute("INSERT INTO Laptop(SerialNumber, " +
            "WeightKg) VALUES(?, ?)", (laptop[0],
            laptop[5]))

with open('PRINTERS.txt') as printers_file:
    printers = csv.reader(printers_file)
    for printer in printers:
        db.execute("INSERT INTO Device(SerialNumber, " +
            "Type, Model, Location, DateOfPurchase, " +
            "WrittenOff) VALUES(?, 'Printer', ?, ?, " +
            "?, ?)", (printer[0], printer[1],
            printer[2], printer[3], printer[4] ==
            'TRUE'))
        db.execute("INSERT INTO Printer(SerialNumber, " +
            "Toner, DateChanged) VALUES(?, ?, ?)",
            (printer[0], printer[5], printer[6]))

db.commit()
db.close()
```

Task	Answer	Marks

4.3	Any 4 marks from:
	Use of `SELECT` … [1]
	… to identify four attributes (SerialNumber, Model, Location, DateCleaned) [1]
	Use of `FROM` with two tables (Device and Monitor) [1]
	… with the serialnumbers compared in the WHERE clause [1]
	Use a (LEFT) INNER JOIN between two (Device and Monitor) table [1]
	… with the serialnumbers compared in the ON Clause [1]
	
	TASK4_3 sample SQL code for query
	
	`SELECT Device.SerialNumber, Device.Model,`
	`       Device.Location, Monitor.DateCleaned`
	`FROM Device, Monitor`
	`WHERE Device.Type = 'Monitor' AND`
	`      Device.SerialNumber = Monitor.SerialNumber;`
		4

4.4	*Mark as follows:* Flask application is run [1] HTML form for entry of `Location` string is implemented [1] Create connection to database [1] SQL query matches `Location` to form input … [1] … AND … [1] … restricts query to devices still in use [1] Display results in HTML table [1] Close connection [1] Browser view of results from query [1] Shows correct results for given input [1] **TASK4_4 possible sample Python code for web application** (see below)	10

```
import flask
import sqlite3

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template('index.html')

@app.route('/filter', methods=['POST'])
def filter():
    db = sqlite3.connect('equipment.db')
    location = flask.request.form['location']
    results = db.execute("SELECT SerialNumber, Type " +
        "FROM Device WHERE Location = ? AND NOT " +
        "WrittenOff", (location,)).fetchall()
    html = flask.render_template('filter.html',
        results=results)
    db.close()
    return html

if __name__ == '__main__':
    app.run()
```

Task	Answer	Marks
Evidence from all tasks	Mark as follows: Some use of indentation and white space (over 50%) [1] Good use of indentation and white space throughout [1] Some evidence of using naming conventions (over 50%) [1] Good evidence of using naming conventions throughout [1] Some evidence of providing comments (over 50%) [1] Thorough clear commenting throughout [1]	**6**