MINISTRY OF EDUCATION, SINGAPORE
in collaboration with
CAMBRIDGE ASSESSMENT INTERNATIONAL EDUCATION
General Certificate of Education Advanced Level
Higher 2

COMPUTING

9569/02

Paper 2 (Lab-based)

October/November 2024

3 hours

Additional Materials: Electronic version of `customerBookings.txt` data file
Electronic version of `firstPage.txt` data file
Electronic version of `villas.txt` data file
Electronic version of `website.txt` data file
Insert Quick Reference Guide

READ THESE INSTRUCTIONS FIRST

Answer **all** questions.

All tasks must be done in the computer laboratory. You are not allowed to bring in or take out any pieces of work or materials on paper or electronic media or in any other form.

Approved calculators are allowed.

Save each task as it is completed.

The use of built-in functions, where appropriate, is allowed for this paper unless stated otherwise.

Note that up to **6** marks out of 100 will be awarded for the use of common coding standards for programming style.

The number of marks is given in brackets [ ] at the end of each question or part question.
The total number of marks for this paper is 100.

This document consists of **13** printed pages, **3** blank pages and **1** Insert.

Singapore Examinations and Assessment Board

Cambridge Assessment International Education

Instruction to candidates:

Your program code and output for each of Task 1 to 4 should be saved in a single `.ipynb` file. For example, your program code and output for Task 1 should be saved as:

`TASK1_<your name>_<centre number>_<index number>.ipynb`

1

Name your Jupyter Notebook as:

`TASK1_<your name>_<centre number>_<index number>.ipynb`

A game designer is designing a game for 2 players and needs you to program a prototype.

Players in the game have 10 randomly generated cards. This is called a 'hand'. The cards each have a colour (red, green or blue) and a number (1, 2, 3 or 4). There is no limit to the number of copies of each card, for example there can be multiple cards that are red and 1. Each player has a score that is 0 when the game starts.

In this prototype, the cards will be randomly generated for each player and sorted into ascending order by number.

On each turn:
- The next card from Player 1's hand is selected.
- The next card from Player 2's hand is selected.
- These 2 cards are compared.

The winning card is determined based on the colour of each card:
- A red card wins over a green card.
- A green card wins over a blue card.
- A blue card wins over a red card.

The number on the winning card is multiplied by 2. This value is then added to the score of the player who had the winning card.

The number on the losing card is added to the score of the player who had the losing card.

If both colours are the same:
- Player 1's card number is added to player 1's score.
- Player 2's card number is added to player 2's score.

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:

```
In [1]: #Task 1.1
        Program code

        Output:
```

Task 1.1

Write a program to set up the game for 2 players by:
- creating a list for each player
- randomly generating 10 cards (each with a colour and number) to each player's list. [4]

Test your program by outputting the contents of each player's hand. [1]

Task 1.2

Write a function that takes the colour of two cards as parameters and returns the result of the comparison. The function should return '1' if the first colour wins, '2' if the second colour wins and '3' if the colours are the same. [2]

Task 1.3

The cards in each hand now need to be sorted into numerical order.

Write a function to take a hand as a parameter, sort the hand in-place into ascending numerical order and return the sorted hand. If two cards have the same number, the colours are compared to position the cards in this order: red, green, blue.

Call the function to sort each of the 2 players' hands. Output the contents of each hand before and after sorting. The contents of a hand must be output on one line.

Do **not** use a built-in sorting function. [8]

Task 1.4

The cards are compared in pairs, starting with the first card from player 1 compared to the first card from player 2.

The appropriate numbers are added to each player's score until all of the 10 cards have been used.

Write a function to calculate and return the final score for each of the 2 players. [5]

Test your program by:
- outputting the sorted hand and the final score for player 1 on one line
- outputting the sorted hand and the final score for player 2 on one line.

[1]

Task 1.5

Write a function to calculate and output which player has won (has the higher final score), and which player has come second.

There must be a suitable output if there is a draw.

All outputs must have an appropriate message. [2]

Test your program. [1]

Save your Jupyter Notebook for Task 1.

2

Name your Jupyter Notebook as:

`TASK2_<your name>_<centre number>_<index number>.ipynb`

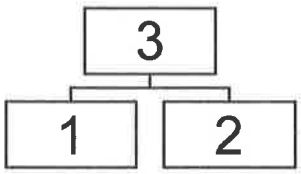
A program is needed to calculate the numbers on each level of a number tree.

The bottom row of the number tree has the numbers 1 to x.

The row above has the result of the addition of the two numbers directly below it. This repeats in pairs until there is only one number left at the top of the number tree. This creates the number tree.

Example 1: This number tree has 2 levels. The first level has the numbers 1 to 2.

The next level adds together these numbers to give 3. There is only 1 number on this level, so the number tree is complete.

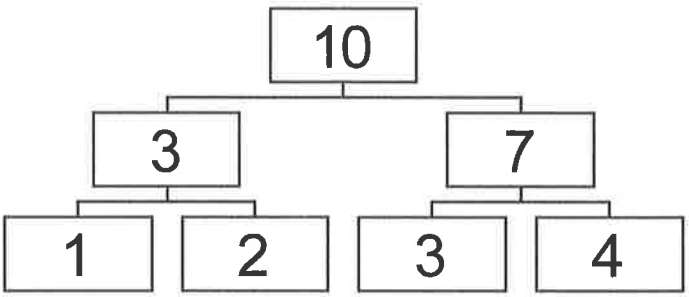


Example 2: This number tree has 3 levels. The first level has the numbers 1 to 4.

The second level adds together these numbers in pairs to give 1 + 2 = 3 and 3 + 4 = 7.

The third level adds together the pair on the second level, 3 + 7 = 10.

There is now only one number at this level, so the number tree is complete.



The number of levels is input into the program.

The program will need to generate and output the number tree for the given number of levels.

The program uses object-oriented programming. Each number in the number tree is a node. The nodes are stored in a list. Each node has the integer number (for example, 1), the index of the left node below it and the index of the right node below it.

The program needs to use local variables throughout.

The class `Node` contains three attributes:

- `data_value` the node's integer data, initialised to –1
- `left_node` the index of the number below it to the left, initialised to –1
- `right_node` the index of the number below it to the right, initialised to –1

–1 represents no data present in `data_value`

–1 represents a null value in a pointer.

The class `Node` has the following methods:
- a constructor that initialises the three attributes
- `set_data()` that assigns a value to `data_value`
- `set_left()` that assigns a value to `left_node`
- `set_right()` that assigns a value to `right_node`
- `get_data()` that returns `data_value`
- `get_left()` that returns `left_node`
- `get_right()` that returns `right_node`

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:

```
In [1]: #Task 2.1
        Program code

        Output:
```

Task 2.1

Write program code to declare the class `Node`, its constructor and its methods. [4]

Task 2.2

This table shows the total number of nodes for number trees with levels 1 to 5.

Number of levels	Total number of nodes
1	1
2	3
3	7
4	15
5	31

The function `calculate_nodes()` takes the number of levels as a parameter. It calculates, outputs and returns the number of nodes for that number tree.

The output must be in an appropriate message, for example:

```
A 3 level number tree has 7 nodes
```

You do not need to validate the parameter. The function must calculate the number of nodes for any number of levels greater than 0.

Write the function `calculate_nodes()`. [3]

Task 2.3

Test your program four times, with the following levels:

```
1
```

```
3
```

```
5
```

```
10
```

Show the output for each test. [2]

Task 2.4

The function `create_empty_tree()` takes 2 parameters; the list of nodes and the number of levels.

The function creates a list of `Node` objects in their initial state with all attributes set to –1. The function returns the list.

The list is zero indexed.

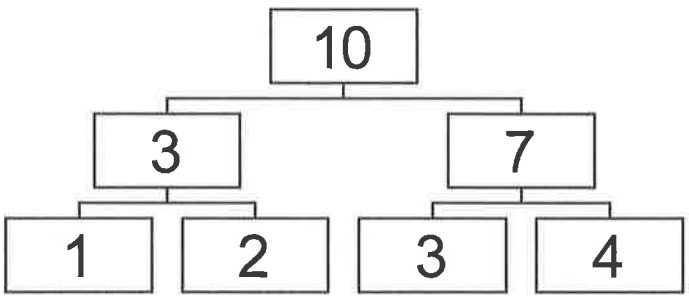
Write the function `create_empty_tree()`. [3]

Task 2.5

The function `assign_leaf_nodes()` sets the data for the lowest level of nodes. The function takes 2 parameters; the list of nodes and the number of levels.

Each leaf node in the first layer is assigned a number starting with 1 in the left-most node and then incrementing by 1 for each subsequent node.

For example, the first level of nodes in this number tree contain the data 1 2 3 4



This table shows the number of leaf nodes for up to 5 levels in a number tree. The function must calculate the number of leaf nodes for any number of levels greater than 0.

Number of levels	Number of leaf nodes
1	1
2	2
3	4
4	8
5	16

For example, if the list of nodes is named `nodes`, then `assign_leaf_nodes(nodes, 3)` would return a list of 7 `Node` objects with the following attributes:

Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6
data_value: 1 left_node: -1 right_node: -1	data_value: 2 left_node: -1 right_node: -1	data_value: 3 left_node: -1 right_node: -1	data_value: 4 left_node: -1 right_node: -1	data_value: -1 left_node: -1 right_node: -1	data_value: -1 left_node: -1 right_node: -1	data_value: -1 left_node: -1 right_node: -1

Write the function `assign_leaf_nodes()`. [3]

Task 2.6

The data for each parent node in the number tree is calculated by adding the data from its two lower nodes.

The function `create_number_tree()`:

- takes 2 parameters: the list of nodes (with the first level of nodes already assigned their data) and the number of levels
- calculates the data for each node in each level in the number tree
- stores the data in each node
- assigns the left pointer of each node to its left lower node
- assigns the right pointer of each node to its right lower node.

For example, if the list of nodes is named `nodes`, then `create_number_tree(nodes, 3)` would return a list of 7 `Node` objects with the following attributes:

data_value: 1 left_node: -1 right_node: -1	data_value: 2 left_node: -1 right_node: -1	data_value: 3 left_node: -1 right_node: -1	data_value: 4 left_node: -1 right_node: -1	data_value: 3 left_node: 0 right_node: 1	data_value: 7 left_node: 2 right_node: 3	data_value: 10 left_node: 4 right_node: 5
Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6

Write the function `create_number_tree()`. [4]

Task 2.7

Amend the program to call the functions `calculate_nodes()`, `create_empty_tree()`, `assign_leaf_nodes()` and `create_number_tree()`.

Output the content of each node in the array in the format:

```
left pointer      data      right pointer
```

[3]

Test your program with levels 1, 3 and 5 and show the output for each test. [2]

Save your Jupyter Notebook for Task 2.

3

Name your Jupyter Notebook as:

`TASK3_<your name>_<centre number>_<index number>.ipynb`

Write a program to format a description for a webpage into an HTML document.

The description for the program will be provided in a text file. Each element will be on a new line and will follow the structure:

`HTML element: content`

For example, in the text file `firstPage.txt` the first line is:

heading: My first page

The HTML element is 'heading' and the text for the heading is 'My first page'.

The HTML elements that can be included are in this table with an example and a description for the content.

HTML Element	Example	Description
heading	`heading: This is a title`	Formats the string `"This is a title"` with the h1 tag
subheading	`subheading: This is a subtitle`	Formats the string `"This is a subtitle"` with the h2 tag
hyperlink	`hyperlink: https://www.seab.gov.sg,Click`	Formats the string `"Click"` as a hyperlink to `"https://www.seab.gov.sg"`
paragraph	`paragraph: This is a paragraph`	Formats the string `"This is a paragraph"` as a paragraph

All documents start and end with the HTML and body tags.

The data being used will not include any special characters. You will not be required to validate the content in each statement, for example that a title is a valid title.

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:

```
In [1]: #Task 3.1
        Program code
```

Output:

Task 3.1

Write **one** function for each of the HTML elements heading, subheading and paragraph to:
- take the text to format as a parameter
- format the text including HTML tags
- return the formatted text.

For example, if the text parameter for the heading function is `"My first page"`, the formatted text will be:

`"<h1>My first page</h1>"` [1]

Test your program by calling the **three** functions with `"My element"` and outputting the return value from each. [1]

Task 3.2

Write a function for the HTML element hyperlink to:
- take the text to format as a parameter
- split the parameter into the URL and string
- format the text including HTML tags
- return the formatted text.

For example, if the text parameter is `"https://www.cambridge.org,Click here"`, the formatted text will be:

`"<a href="https://www.cambridge.org">Click here</a>"` [1]

Test your program by calling the function with `"https://www.cambridgeinternational.org, Click here"` and outputting the return value. [1]

Task 3.3

The function `body()` takes a list as a parameter containing the text to convert into an HTML formatted document. Each element in the list has one HTML element for conversion.

Write the function to:
- call the appropriate function for each HTML element in the list
- create a single string containing all the formatted returned HTML elements
- insert the HTML and body opening and closing tags at the start and end of the string
- return either the complete formatted string, or `False` if there are any unrecognised elements. [6]

Task 3.4

The text file `firstPage.txt` contains the description of a web page that needs converting into an HTML document.

Write the function `read_data()`, to read in the data from a text file. The function needs to:
- take the name of a file (including extension) as input
- read each line from the file into a separate list element
- call the appropriate functions to convert the list to a formatted HTML string
- return the formatted HTML string.

Write the function `write_data()`, to store the formatted HTML string as an HTML file. The function needs to:
- take the name of a file (excluding extension) as input
- take the formatted HTML string to write as input
- insert the string `<!doctype html>` at the start of the file
- write the formatted string to the file with the name input and the extension `".html"`.

Output an appropriate message if the data cannot be converted into HTML.

Call the function `read_data()` then call the function `write_data()`. [9]

Run your program **two** times with the following inputs:

Test 1:

```
firstPage.txt

test
```

Test 2:

```
website.txt

theWebsite
```

[3]

Save your Jupyter Notebook for Task 3.

4

Name your Jupyter Notebook as:

`TASK4_<your name>_<centre number>_<index number>.ipynb`

A holiday company has 10 villas that customers can hire. A program is being created to store the customer bookings and allow the company staff to check the availability of the villas on requested dates.

The text files `customerBookings.txt` and `villas.txt` store the current data.

The data in `villas.txt` is stored in the following format:

villa ID, villa name, country, cost

The data in `customerBookings.txt` is stored in the following format:

booking ID, customer ID, villa ID, start date, number of days

You do not need to consider how data about the customers will be stored.

You can assume the year is 2025.

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:

```
In [1]:  #Task 4.1
         Program code

         Output:
```

Task 4.1

Write a Python program that uses SQL to create a database with two tables: one table to store data about the villas and one table to store data about customer bookings.

Define the primary and foreign keys for each table. [7]

Task 4.2

Write a Python program that uses SQL to insert the data from each of the two text files into the appropriate field in each database table. [4]

Run your program to test the data has been entered correctly. [1]

Task 4.3

A new table `Villa_Booking`, needs creating to store every date that each villa has been booked. This will have two fields: one for the villa ID and one for the date.

For example, if villa number 1 is booked from the 01-Jan for 4 days, the following data will be inserted into `Villa_Booking`:

```
1   01-Jan

1   02-Jan

1   03-Jan

1   04-Jan
```

Write a Python program that uses SQL to create `Villa_Booking` and populate it with the existing customer bookings. [5]

Task 4.4

Write a Python function to:

- take as input the villa name, start date and number of days a customer would like to book
- use SQL to check the availability of the villa for the customer's request
- output a list of all the requested dates that the villa is available
- output a list of all the requested dates that the villa is not available.

All outputs must have an appropriate message. [6]

Test your program with the following data:

Villa name	`Dolphin`
Month	`Apr`
Date	`8`
Number of days	`4`

[1]

Save your Jupyter Notebook for Task 4.

BLANK PAGE

BLANK PAGE

BLANK PAGE