ANGLO-CHINESE JUNIOR COLLEGE
JC2 PRELIMINARY EXAMINATION

Higher 2

COMPUTING 9569/02

Paper 2 (Lab-based) **7 August 2023**

3 hours

Additional Materials: Electronic version of `Task2.txt` data file
Electronic version of `numbers.txt` data file
Electronic version of `club.db` data file
Electronic version of `DATA.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.

Instruction to candidates:

Your program code and output for each of Tasks 1, 2 and 3 should be downloaded in a single `.ipynb` file. For example, your program code and output for Task 1 should be downloaded as

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

The task is to write addition and multiplication algorithms for large integers, assuming that the results of adding and multiplying single-digit integers have already been hard-coded.

The addition algorithm described below is similar to what students learn in primary school.

1. Declare and initialise variables:
   - Create two string variables, `s1` and `s2`, to store the numbers to be added;
   - The integer `carry` initially has a value of 0;
   - The string `result` initially is an empty string.

2. If `s1` and `s2` are of unequal length, pad the shorter string with `'0'`s in front to make the two strings the same length.
   For example, if `s1` is `'1238'` and `s2` is `'15'`, pad `s2` so that `s2` becomes `'0015'`.

3. Create a loop to iterate over both strings, starting from the largest index. For each cycle of the loop, perform the following steps.
   - Retrieve the digits of `s1` and `s2` at that index. Call them `digit1` and `digit2` respectively.
   - Define `column_sum` to be the numerical sum of `digit1`, `digit2` and `carry`.
   - Concatenate the last (ones) digit of `column_sum` to the front of `result`. This is the new value of `result`.
   - The first (tens) digit of `column_sum` is the new value of `carry`.

4. If `carry` is larger than 0, concatenate `carry` to the front of `result`. This is the new value of `result`.

5. Return `result`.

The Karatsuba algorithm is a recursive method of multiplying large (positive) integers efficiently. The recursive algorithm described below is a simplified version. The function `karatsuba(s1,s2)` takes two string variables, `s1` and `s2`, which consist entirely of digits, and returns an integer.

1. If `s1` and `s2` are both of length 1, multiply them as single-digit numbers and return their product.

2. If `s1` and `s2` are of unequal length, pad the shorter string with `'0'`s in front to make the two strings the same length. For example, if `s1` is `'1238'` and `s2` is `'15'`, pad `s2` so that `s2` becomes `'0015'`.

3. Define the following variables.
   - `leng ← LENGTH(s1)`
   - `mid ← length DIV 2`
   - `a ← LEFT(s1, leng - mid)`
   - `b ← RIGHT(s1, mid)`
   - `c ← LEFT(s2, leng - mid)`
   - `d ← RIGHT(s2, mid)`

4. Use the `karatsuba` function recursively to find the following variables.
   - `ac ← STRING(karatsuba(a,c))`
   - `ad ← STRING(karatsuba(a,d))`
   - `bc ← STRING(karatsuba(b,c))`
   - `bd ← STRING(karatsuba(b,d))`

5. Perform the following concatenations.
   - (`2 * mid`) copies of `'0'` to the end of `ac`
   - `mid` copies of `'0'` to the end of `ad`
   - `mid` copies of `'0'` to the end of `bc`

6. Convert `ac`, `ad`, `bc` and `bd` to integers and add them. Return the sum.

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 program code to define `addition(s1,s2)` that implements the algorithm to sum two numbers represented as strings together. The function would return an integer. You do not need to perform data validation.

For this sub-task, you are allowed to use the `+` operator only to add single-digit numbers together, and to concatenate strings. [5]

Task 1.2

Test your code in **Task 1.1** with the following test cases.

- `print(addition('12','34') == 46)`
- `print(addition('1234','222') == 1456)`
- `print(addition('999','1') == 1000)` [2]

Task 1.3

Write program code to define `karatsuba(s1,s2)`. You do not need to perform data validation.

For this sub-task, you are allowed to use the `*` operator only to multiply single-digit numbers together, and to concatenate multiple copies of strings. [7]

Task 1.4

Test your code in **Task 1.3** with the following test cases.

- `print(karatsuba('15','20') == 300)`
- `print(karatsuba('1234','22') == 27148)`
- `print(karatsuba('999','222') == 221778)` [2]

Task 1.5

Write program code to:

- Initialise a list `lst1`.
- Randomly generate 50 integers in the range 1 to 100 (inclusive) and store them in `lst1`.
- Find the sum of all the numbers in `lst1` with the `addition` function from **Task 1.1** and store the result as `total_sum`. Display `total_sum`.
- Find the product of all the numbers in `lst1` with the `karatsuba` function from **Task 1.3** and store the result as `total_prod`. Display `total_prod`.

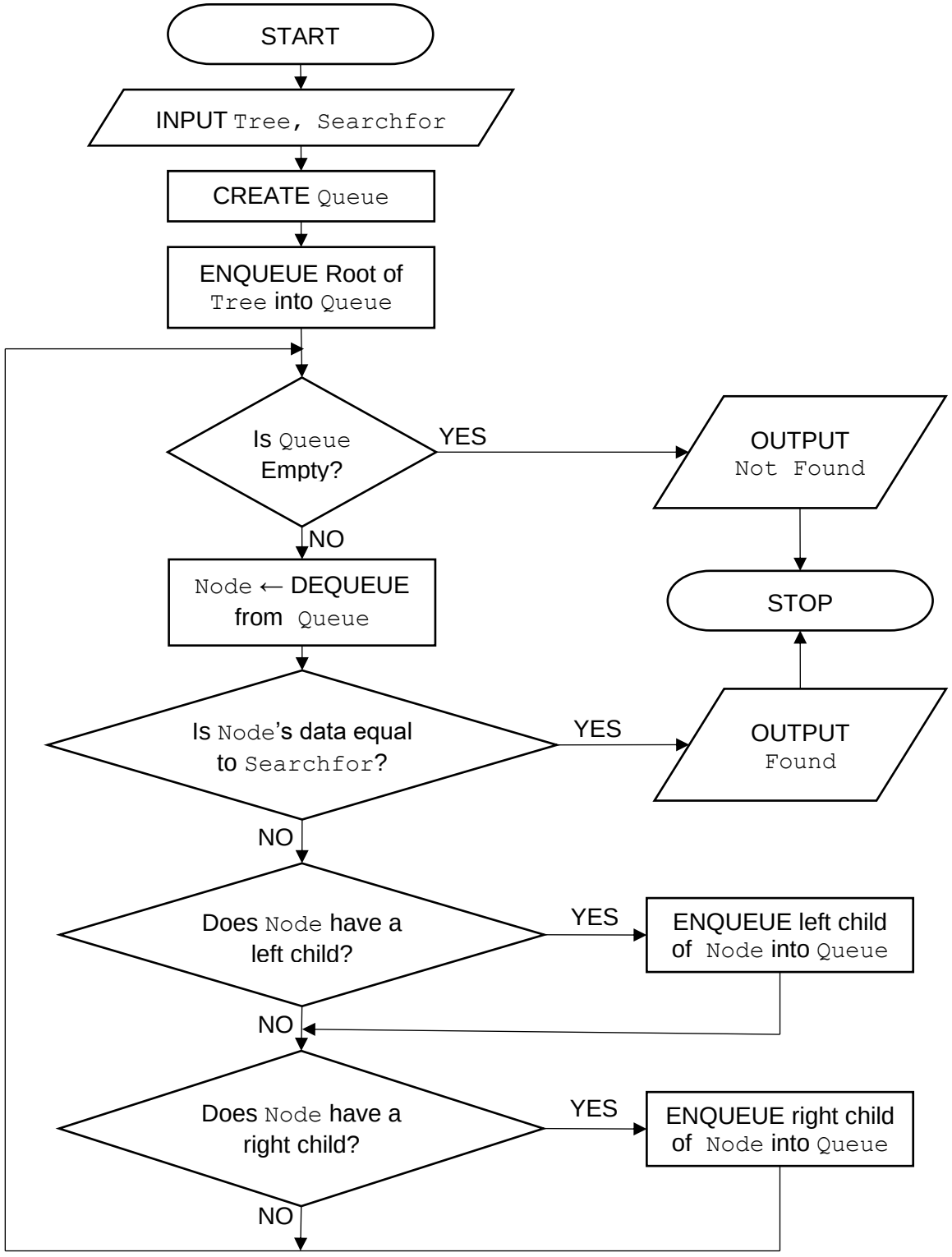
For this sub-task, you are not allowed to use the `+` and `*` operators. [4]

Save your Jupyter Notebook for Task 1.

2 Name your Jupyter Notebook as:

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

The task is to use a circular queue to perform a breadth-first search through a binary tree, by following the algorithm in the flowchart below. This is done by using Object-Oriented Programming (OOP) to create a circular queue inside an Array, and to create the nodes for a binary tree data structure.



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

Define an `Array` class with the following properties:

- `capacity`: an integer which gives the size of the array;
- `container`: a list of length `capacity` which is initialised with all entries as `None`

and the following methods:

- `add_data(new_data, i)` replaces the entry in index `i` of `container` with `new_data`;
- `get_data(i)` returns the entry in index `i` of `container`;
- `delete_data(i)` replaces the entry in index `i` of `container` with None. [6]

We use an array to hold a circular queue. The head pointer of the circular queue is the index of the first item of the circular queue, and the tail pointer of the circular queue is the index of the first `None` entry in the array after the circular queue. The queue is full when there is only one `None` entry in the array.

Define `CircQueue` as a subclass of `Array` with the following additional properties:

- `head`: the value of the head pointer, initialised as `0`;
- `tail`: the value of the tail pointer, initialised as `0`

and the following additional methods:

- `enqueue(new_data)` enqueues `new_data` into the circular queue if the circular queue is not full, and outputs an appropriate error message otherwise;
- `dequeue` removes the item at the head of the circular queue, and returns it;
- `get_head` returns the value of the head pointer. [11]

Task 2.2

Define a `Node` class with the following properties:

- `data`: the data of the node;
- `left`: the left child of the node, initialised as `None`;
- `right`: the right child of the node, initialised as `None`

and the following methods:

- `make_left_child(lchild)` makes `lchild`, another `Node` object, the left child of this node
- `make_right_child(rchild)` makes `rchild`, another `Node` object, the right child of this node
- `get_data` returns the data of the node
- `get_left_child` returns the left child of the node
- `get_right_child` returns the right child of the node. [4]

Task 2.3

Create a tree by copying and pasting the contents of `Task2.txt` into this Jupyter Notebook.

Write program code to:

- ask the user for a value to search for;
- create `Q`, a `CircQueue` object with a `capacity` of 10;
- use `Q` to perform a breadth-first search through the tree created previously;
- give appropriate output depending on whether the user's entered value is found in the tree.

[11]

Save your Jupyter Notebook for Task 2.

3 Name your Jupyter Notebook as:

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

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 a function `insertion_sort(lis)` that sorts the items of `lis`, which are all positive integers, in place using an insertion sort algorithm. It is not necessary to perform data validation. [4]

Task 3.2

Write program code to:

- read the contents of `numbers.txt`;
- sort the numbers using the insertion sort algorithm in **Task 3.1**;
- write the sorted list to a new file, `sorted.txt`, with one integer per line. [6]

Task 3.3

The ancient Egyptians used hieroglyphic symbols to write numbers.

Each power of 10 from 1 to 10000 had its own symbol, according to the table below.

Number	1	10	100	1000	10000
Eqyptian hieroglyphic symbol					
Unicode value	78842	78726	78690	78268	77997

To represent a number, each of the symbols was repeated as many times as necessary. For example,

729 = 700 + 20 + 9

would be written as

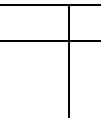
If one of the digits is 0, no symbols are used. For example,

709

would be written as

To accommodate this in Unicode, each set of repeated symbols is encoded as a single character. These occupy consecutive numbers in the Unicode system. For example, the multiples of 100 are show below.

Number	100	200	300	400	500
Eqyptian hieroglyphic symbol					
Unicode value	78690	78691	78692	78693	78694

Number	600	700	800	900
Eqyptian hieroglyphic symbol				
Unicode value	78695	78696	78697	78698

Write program code to:

- convert the numbers in the sorted list from **Task 3.2** into Egyptian hieroglyphic symbols;
- show the output in Jupyter Notebook. [8]

Save your Jupyter Notebook for Task 3.

4 A karate club stores information about its members. This includes both personal information, as well as information about competitions they have participated in and their results.

The following is an example of information stored in the table.

Mem_ID	Mem_Name	Mem_DOB	Comp_Name	Comp_ID	Comp_Score
M01	John Tan	19961224	World Karate Meet	C01	250
M01	John Tan	19961224	Asean Karate Meet	C03	170
M01	John Tan	19961224	Cola Karate Championship	C09	300
F03	Susan Lim	20110924	Cola Karate Championship	C09	190
F03	Susan Lim	20110924	World Karate Meet	C01	260

Task 4.1

Create an SQL file called `task4_1.sql` to show the SQL code to create a database `club.db` with three tables: Members, Competitions, and Scores. The primary and foreign keys for each table should be defined.

Save your SQL file as
`Task4_1_<your name>_<centre number>_<index number>.sql` [6]

Task 4.2

The file `DATA.txt` contains the comma-separated values to be stored in the database.

Write a Python program to read the data from the file and then store it in the database.

Save your Python program as
`Task4_2_<your name>_<centre number>_<index number>.py` [6]

Task 4.3

Write a Python program and the necessary files to create a web application that allows a competition organiser to record the scores of the competition's participants.

The organiser should be able to enter the following input, and the form would store it in the correct place in the database.

- Competition ID
- Member ID
- Score

You may assume the user would only enter valid Competition IDs and participant IDs.

Save your program code as
`TASK4_3_<your name>_<centre number>_ <index number>.py`

with any additional files/subfolders as needed in a folder named
`TASK4_3_<your name>_<centre number>_ <index number>` [6]

Task 4.4

Write a Python program and the necessary files to create a web application that, given a Competition ID, displays the following for each competitor in that competition

- Name;
- Member ID;
- Scores.

The data should be presented in a table sorted by descending order of Scores

Save your program code as
`TASK4_4_<your name>_<centre number>_ <index number>.py`

with any additional files/subfolders as needed in a folder named
`TASK4_4_<your name>_<centre number>_ <index number>` [5]

Run the web application with the following input:

- Competition ID: `C02`

and save the output of the program as
`TASK4_4_<your name>_<centre number>_ <index number>.html` [1]