ANGLO-CHINESE JUNIOR COLLEGE
JC1 PROMOTIONAL EXAMINATION

Higher 2

COMPUTING 9569/02

Paper 2 (Lab-based) **25 August 2023**

1.5 hours

Additional Materials: Electronic version of `primelist.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 **3** marks out of 50 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 50.

Instruction to candidates:

Your program code and output for each of Tasks 1 and 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`

A **prime number** is a positive integer *p* that does not have any positive divisors other than 1 and *p* itself.

- By convention, 1 is not a prime number.
- 5 is a prime number as its only divisors are 1 and 5.
- 6 is not a prime number as 2 and 3 are divisors of 6.

The 17th century French priest Marin Mersenne was interested in prime numbers which are of the form $2^n - 1$, where *n* is a positive integer. These numbers are now known as Mersenne primes. The mathematicians Édouard Lucas and Derrick Lehmer discovered a fast method for determining whether a Mersenne prime $2^n - 1$ is a prime number.

First, check if *n* itself is a prime number. If it is not, then $2^n - 1$ is not a prime number.

If *n* is a prime number, calculate u(*n*) is using the flowchart on Page 3. If u(*n*) is 0 then $2^n - 1$ is a prime number. Otherwise, $2^n - 1$ is not a prime number.

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 the function `isprime(p)` that determines whether `p` is a prime number. The function also needs to validate that `p` is a positive integer. [4]

Task 1.2

Write programme code to define the function `mersenne(n)` that determines whether $2^n - 1$ is a prime number using Lucas and Lehmer's method. You do not need to perform data validation. [6]

Task 1.3

Test the functions written in **Task 1.1** and **Task 1.2** using the following test cases.

- $2^7 - 1$ is a prime number.
- $2^{11} - 1$ is not a prime number. [4]

Task 1.4

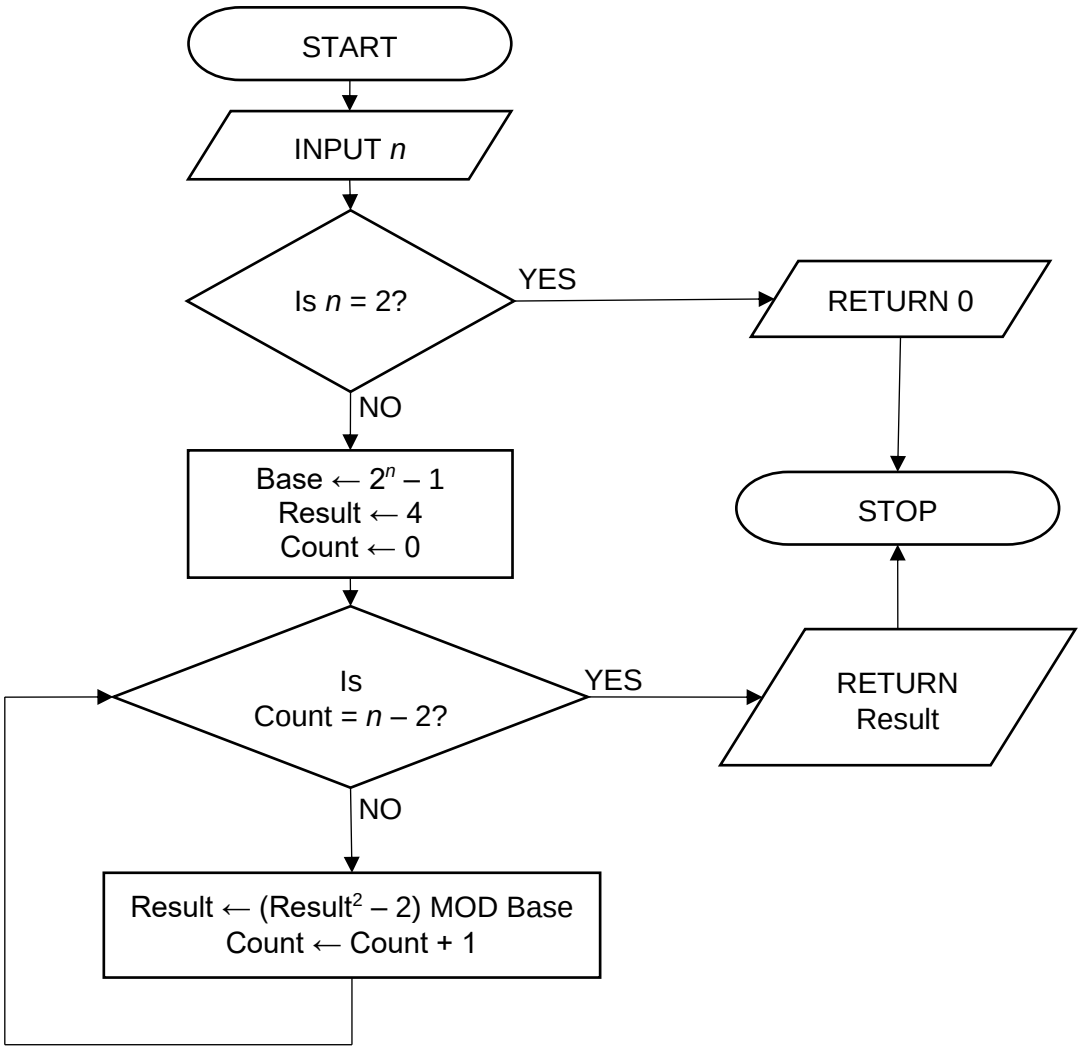
Mersenne gave a list of numbers *n* for which he claimed that $2^n - 1$ is a prime number. This list is reproduced in `primelist.txt`.

Write code to read `primelist.txt` and generate the following two files.

- The file `missedlist.txt` contains positive integers *n* up to 300 for which $2^n - 1$ is a prime number, but were missed out in Mersenne's list.
- The file `wronglist.txt` contains numbers *n* in Mersenne's list for which $2^n - 1$ is actually not a prime number. [9]

Save your Jupyter Notebook for Task 1.

Flowchart to calculate u(*n*):



2 Name your Jupyter Notebook as:

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

A linked list is a simple data structure that can form the basis of more complicated data structures. The table below shows the properties and methods of a `LinkedList` class and a `Node` class.

Node
- data: INTEGER - pointer

The pointer attribute points to the following node in the linked list.

LinkedList
- head
+ Constructor() + add_head() + add_index() + remove_head() + remove_index() + display()

- The head attribute points to the node at the head of the linked list.
- `add_index(index,n)` adds a new node, whose data is n, into the linked list at a the location specified by index. The node at the head of the linked list has index 0. This method should return an error message if index is out of range.
- `remove_index(index)` removes the node from the linked list object at the location specified by index. This method should return an error message if index is out of range.
- `display()` outputs all the data stored in the nodes in the linked list, in order.

A double-ended queue differs from a regular linear queue in the following ways:

- Data can be added at both ends of the double-ended queue.
- Data can be removed from both ends of the double-ended queue.

As a result, a double ended queue does not follow First In First Out (FIFO).

The `DEQueue` (double-ended queue) class can be implemented as a subclass of the LinkedList class. It would have the following properties and methods.

DEQueue
- head - tail
+ Constructor() + add_head(data) + add_tail(data) + remove_head() + remove_tail() + display()

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 for the `Node` and `LinkedList` classes. [13]

Task 2.2

Write program code for the `DEQueue` class, using inheritance appropriately. [8]

Task 2.3

Write program code to:

- Declare a new instance of `DEQueue`.
- Add the integer 1 to the head of the `DEQueue` object.
- Add the integer 2 to the tail of the `DEQueue` object.
- Call `remove_head()`.
- Add the integer 3 to the tail of the `DEQueue` object.
- Call `remove_tail()`.
- Add the integer 4 to the tail of the `DEQueue` object.
- Display the contents of the `DEQueue` object. [3]

Save your Jupyter Notebook for Task 2.