**NJC 2017 P1Q1 [15 marks]**

**1** The head of the mathematics department would like to combine and rank the examination results from 3 classes so that a summary report can be generated. The mathematics teachers from these classes were instructed to use the following format (on each line of a text file) when collating the results:

<student index number>,<student examination score>

After the results from each class had been collated, the following was discovered. Each <student index number> was stored as a denary value. However, the mathematics teachers stored <student examination score> values using different base number systems.

The following were the base number systems utilised:

- Class 2A: binary
- Class 2B: octal
- Class 2C: hexadecimal

It should be noted that all index numbers and examination scores correspond to non-negative integer values.

**Task 1.1**

Write the program code for the function `base_n_to_denary(value_str, n)`, which converts the given base `n`, non-negative integer value, `value_str`, into its corresponding denary value, and then returns it.

More specifically, this function should take the following arguments.

- `value_str`: a string corresponding to a base `n` value.
- `n`: a positive integer between 2 and 16 corresponding to the base number system used to represent `value_str`.

The function should then return an integer (i.e., a denary value), whose value is equal to `value_str`. You may not use any inbuilt functionality to perform this conversion.

**Evidence 1**

- The program code for the function `base_n_to_denary(value_str, n)`. [3]

As mentioned above, the head of the mathematics department would like to generate a summary report of the examination.

The format of this summary report is as follows:

```
Mathematics results for classes 2A, 2B and 2C:

The highest examination score: 100.0
The average examination score: 50.0
The lowest examination score: 0.0

The top 3 students are:

  Class      Index       Mark
    2A         1         100
    2C         10         98
    2B         3          96
```

Note that the values used in the example summary report above are not based on the actual values that are found in the given files.

**Task 1.2**

Write the program code to perform the following.

- Read and store the contents of the files:
  - `2A_SCORES.TXT`
  - `2B_SCORES.TXT`
  - `2C_SCORES.TXT`
- When storing this data, utilise the function `base_n_to_denary(value_str, n)` to convert the scores in each file to their corresponding denary values.

**Evidence 2**

- The program code to perform the above task. [3]

**Task 1.3**

Write the program code for the function `sort_student_scores(...)`, which uses the class, index and score data (i.e., the data retrieved from the 3 files listed in **Task 1.2**), and returns the same information, but sorted in **descending order** based on score.

When considering the parameters and return value for `sort_student_scores(...)`, your decision(s) should be based upon the usage of this information to produce the summary report described earlier in this question.

Note that you are to utilise the **quicksort** algorithm to perform the required sorting. Also note that you may not utilise any inbuilt functionality to search or sort when writing this function.

**Evidence 3**

- The program code for the function `sort_student_scores(...)`. [5]

**Task 1.4**

Utilise the data from **Task 1.2**, and the function implemented in **Task 1.3** – i.e., `sort_student_scores(...)`, to produce the summary report described earlier in this question.

Note that you may not use any inbuilt functionality to find the:

- Average score
- Maximum score
- Minimum score

**Evidence 4**

- The program code to print the summary report. [3]

**Evidence 5**

- A screenshot of the summary report output. [1]

**HCI 2020 MYE P2Q2 [15 marks]**

2. A prime number is an integer greater than 1 that is only divisible by 1 and itself. For example, the first five prime numbers are 2, 3, 5, 7 and 11.

The task is to generate all prime numbers and display the total number of prime numbers up to a specified number.
For each of the subtasks, 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
```

In [2]:

```
# Task 2.2
Program Code
```

Output:

In [3]:

```
# Task 2.3
Program Code
```

Output:

**Task 2.1**

Write program code for a function to determine whether a given integer `n` is prime. Use the following specification.

```
        FUNCTION isPrime(n: INTEGER) RETURNS BOOLEAN
```

The function has a single parameter `n` and returns `TRUE` if `n` is prime, and `FALSE` otherwise.

[4]

**Task 2.2**

Write program code to:

- use the `isPrime()` in Task 2.1
- display all the prime numbers up to 100
- print out the total number of prime numbers up to 100. [4]

The most efficient way to find all the prime numbers less than or equal to an integer value `n` is by using **The Sieve of Eratosthenes**. It was developed by Eratosthenes, an ancient mathematician.

The algorithm is as follows:

- Write down all numbers from 0 to `n`
- Cross out 0 and 1 because they are not prime
- Set `p` equal to 2
- **While** `p` is less than `n` **do**
    - Cross out all multiples of `p` (but not `p` itself)
    - Set `p` to the next number in the list that is not crossed out
- Output all of the numbers that have not been crossed out as prime

<u>Example</u>

To find all the prime numbers less than or equal to n, say 20.

Generate a list of integers from 0 to 20

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |

Cross out 0 and 1

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ~~0~~ | ~~1~~ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |

Cross out all multiples of 2

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | ~~4~~ | 5 | ~~6~~ | 7 | ~~8~~ | 9 | ~~10~~ | 11 | ~~12~~ | 13 | ~~14~~ | 15 | ~~16~~ | 17 | ~~18~~ | 19 | ~~20~~ |

Cross out all multiples of 3

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | | 5 | | 7 | | ~~9~~ | | 11 | | 13 | | ~~15~~ | | 17 | | 19 | |

5, 7, 11, 13, 17 and 19 contain no multiples in the list

The numbers not crossed out at this point, [2, 3, 5, 7, 11, 13, 17, 19], are all the prime numbers less than or equal to 20

**Hint: You may simulate crossing out a number by replacing it with 0. Then, once the algorithm completes, all of the non-zero values in the list are prime.**

**Task 2.3**

Write program code to:
- implement this algorithm
- display all the prime numbers between 2 and 100. [7]

Download your program code and output for Task 2 as
`TASK2_<your name>_<ct>.ipynb`

**DHS 2015 P1Q3 [41 marks]**

**3.** International Standard Book Number (ISBN) is a unique number assigned to each edition of a book. It has two formats: ISBN-10 and ISBN-13. The former is used before 1 Jan 2007 and the latter after that. Examples of valid ISBNs are 020103803X and 978-1-284-05591-7. The last digit is the check digit and is computed as follows:

**ISBN check digit (10 digits) - mod 11 algorithm**

- Each digit starting from left to right is assigned a weight from 1 to 9. Each digit is multiplied by its position weight. The sum of products modulo 11 gives a remainder between 0 and 10. If the remainder is 10, the check digit is the roman numeral X, else the check digit is the remainder.
- **Example ISBN-10: 0-07-063546-3**
- (0x1) + (0x2) + (7x3) + (0x4) + (6x5) + (3x6) + (5x7) + (4x8) + (6x9) = 190
- 190 mod 11 = 3
- Hence 3 is the check digit.

**ISBN check digit (13 digits) - mod 10 algorithm**

- Each digit starting from the left to right is multiplied by 1 or 3 alternatively. The sum of the products modulo 10 gives a remainder between 0 to 9. If the remainder is non-zero, subtract this from 10 to get the check digit, else the check digit is 0.
- **Example ISBN-13: 978-0-07-063546-3**
  1x9 + 3x7 + 1x8 + 3x0 + 1x0 + 3x7 + 1x0 + 3x6 + 1x3 + 3x5 + 1x4 + 3x6 = 117
- 117 mod 10 = 7
- 10 - 7 = 3
- Hence 3 is the check digit.

**Task 3.1**
Write a function `ISBN_Check_Digit(isbn)` which generates the check digit for a given ISBN-10 or ISBN-13 number.

**Evidence 9**
Program code. [5]

**Evidence 10**
Screenshots (1 for ISBN-10 and 1 for ISBN-13) [2]

**Task 3.2**
Write a function `Valid_ISBN(isbn)` which calls your `ISBN_Check_Digit(isbn)` function in Task 3.1 to determine if a given ISBN-10 or ISBN-13 number is valid.

**Evidence 11**
Program code. [4]

**Evidence 12**
Screenshots for appropriately generated ISBN-10 and ISBN-13 numbers. [2]

An ISBN-10 number can be converted to its ISBN-13 equivalent by prefixing '978' to the ISBN-10 number and then calculating the check digit using the ISBN-13 algorithm.

**Task 3.3**
Write a function `ISBN10_To_ISBN13(isbn)` to convert an ISBN-10 number to its ISBN-13 equivalent.

**Evidence 13**
Program code. [3]

**Evidence 14**
Screenshot of output for ISBN-10 number 1904467520. [1]

**Task 3.4**
Write a function `ISBN13_To_ISBN10(isbn)` to convert an ISBN-13 number to its ISBN-10 equivalent.

**Evidence 15**
Program code. [4]

**Evidence 16**
Screenshot of output for ISBN-13 number 9780748740468. [1]

A small library wishes to store its book collection details using a random file organisation. It currently holds 200 books but wishes to expand its collection by about 10% every year. To resolve collisions, it decides to use double hashing which minimises clustering. For a start, assume that this file organisation will be maintained for 3 years.

**Task 3.5**
Devise and implement a suitable double hashing strategy `Hash_Key(isbn)`. Annotate your strategy using program comments.

**Evidence 17**
Program code with comments. [5]

**Task 3.6**
Write program code to generate an appropriate number of ISBNs to the random text file `LIBRARY.txt` which will cater to the library's collection growth over 3 years. Include `Insert_Book(isbn)` and `Lookup_Book(isbn)` functions to insert a book and lookup a book respectively.

**Evidence 18**
Program code. [8]

**Evidence 19**
Screenshots showing the insertion and lookup of one non-collided record and one collided record. [4]

**Evidence 20**
Separate printout of `LIBRARY.txt` after insertion of all generated records. [2]

**NYJC 2020 MYE P2Q3 [19 marks]**

**3** You are to design an object-oriented program which simulates a print queue for a printer on a local area network (LAN). The print queue consists at any time of none, one, or more print jobs.

Each user can send a print job from any of the terminals on the LAN. Each terminal on the network is identified by an integer number in the range 1 to 200.

The program you are to design will record for each print job:

- the user ID
- the terminal number from which the print request was sent
- the job type - `'C'` for colour, `'B'` for black and white
- the file size (integer in Kbytes)

In practice, there are several print queues each associated with a different printer. Each printer is identified by a short name such as `Comp_Lab2`.

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

In [2]:   #Task 3.2
          Program code

In [3]:   #Task 3.3
          Program code

In [4]:   #Task 3.4
          Program code

           Output:
```

**Task 3.1**

Design and write program code to define one or more classes and other appropriate data structures for this application. [6]

A print queue behaves as a queue data structure.

Assume, for testing purposes:

- there is a single printer on the LAN.
- the maximum print queue size for the printer is ten print jobs.

The main program will simulate:

- the sending of print jobs to the printer by different users.
    - that is, the addition of a print job to the print queue.
- the output of a job from the print queue.
    - that is, the removal of a print job from the print queue.

The program design has the following menu:

```
1. Add print job to print queue
2. Output next print job from printer
3. Display current print queue (all jobs in queue)
4. End simulation
```

The program simulates the working of the print queue as follows:

1. The empty print queue is initialized.
2. The program user selects menu options 1, 2 and 3 in any order.
3. The program user selects menu option 4.

**Task 3.2**

Write program code to:

- display the main menu.
- input the choice by the user.
- run the appropriate code for the choice made. [3]

**Task 3.3**

Write program code to initialize the print queue for `Comp_Lab2` printer.

Write program code to display the current state of the print queue. [4]

**Task 3.4**

Write program code to add a new print job to the print queue. The requirement will be:

- program user enters data for the new print job.
- print job is added to the print queue.

Write program code to output the next print job from the printer. This code will execute from menu option 2. Test your program by:

- adding three print jobs
- outputting the next print job. [6]

Save your program code and output for Task 3 as
`TASK3_<your name>.ipynb`

**HCI 2020 BE2 P2Q1 [10 marks]**

1. The task is to create a NoSQL database for a library to manage its book records.

   **Task 1.1**

   Write a python program to:

   - Create a MongoDB database named `library` and a new collection named `books`.
   - Insert the book documents into the `books` collection in the `library` database. Use the sample dataset given in the file `BOOK.txt`. You should paste the contents of this file into your program.
   - Display all book documents in the `books` collection.

   Save your program code as `TASK1_1_<your name>_<ct>.py`

   Run the program. Produce a screenshot of the output and save it as
   `TASK1_1_<your name>_<ct>.jpg` [3]

   **Task 1.2**

   Write program code which make use of `books` collection in `library` database to:

   - display the `book id`, `title` and `author` for those books published in 2015.
   - display all book documents with `page_count` greater than or equal to 100 and less than 400.
   - update `page_count` field to `'Less Than 100 Pages'` for those books where `page_count` field does not exist.
   - display the `book title`, `page_count` and `year` of publication according to the `year` in descending order.

   **All outputs should have appropriate messages to indicate what you are showing.**

   Save your program code as `TASK1_2_<your name>_<ct>.py`

   Produce a screenshot of the output and save it as
   `TASK1_2_<your name>_<ct>.jpg` [7]