Task 1.1

For reading data from file

- Opens and closes the file for reading [1]
    - extracting only the ID number from each Row from file [1]
- Fixed array size of 20 with placeholder `None` values [1]
- Calculation of `Index` based on given formula [1]
- Assigns values ID to the correct index in the array [1]
- Uses loop to repeat until end of file [1]
- Contents of array printed in the correct order with or without the placeholder values [1]

Task 1.2

- Check if data exists in the index location in the array [1]
- Array is searched sequentially from the conflicted index location, to find [1]
    - index of empty location [1]
- Contents of array printed in the correct order with or without the placeholder values [1]

Task 1.3

- Creates procedure for searching the array obtained in Task 1.2 [1]
    - Uses loop to pass through array values to check if the values exists in the array and get the index [1]
    - Explicit handling of the case where value is not found [1]
- Evidence of testing [1]
- Test result is accurate. 1 for each correct one [3]

Task 4.1

- The class is called `Book` [1]
- The class offers a implementations of
    - `summary` [1]
    - at least a pair of `get` and `set` `id`, `title` and `price` [1]
        - **all** 3 `get` and `set` `id`, `title` and `price` correct [1]
- The constructor that takes `id`, `title` and `price` [1]

Task 4.2

- Implementation of `compare_with()` which takes a `Book` item and parameters *t* or *p* for basis of comparison... [1]
  - . . . use `if` to handle the two different cases of options passed in [1]
  - . . . compares incoming parameter's data with member data [1]
- Creation of an empty list variable to fill in Book objects [1]
- Correctly including the objects as specified in the text file [1]
- Creating an object from the text in the text file [1]
- Creating all four objects [1]
- Evidence of using the `compare_with()` method to determine the position in the list [1]
- Iteration to create list with several items [1]
- Iterating over the list and calling the summary method for each item [1]
- four summary lines [1]
- four descriptions for the book items in the correct order by alphabetical order [1]
- four descriptions for the book items in the correct order by price [1]

Task 4.3

- The classes are called `ElectronicBook` and `AudioBook` [1]
- The constructors should call the base constructor [1]
- The `summary()` methods should call or overwrite the base `summary()` method. . . [1]
  - . . . and also return the pages or durations to the summary [1]
- Implementation of `compare_with()` should take the `ElectronicBook` and `AudioBook` item. . . [1]
  - . . . and incoming parameter's type should be checked [1]
  - . . . and `ElectronicBook` items should be handled separately from `AudioBook` items [1]
- Correctly adding each defined objects with its correct type [1]
- Nine summary lines [1]

- Five Electronic Book followed by four Audiobook [1]
- Five Electronic Book in the correct order [1]
- Four Audio Book in their correct order [1]

Task 3.1

- ask for input, function or otherwise [1]
- validity check on the input,... [1]
  - ...which checks that characters belong in `0123456789ABCDEF` [1]
- Uses loop to pass through input string to... [1]
  - ...convert the hexadecimal char into its denary form via dictionary, array or otherwise [1]
- multiplying the string with 16 [1]
  - of correct powers [1]
- keeping a identifier to keep track of the total, e.g. `tot` [1]
- correct conversion for valid inputs [1]

Task 3.2

- Opens and closes the file for reading [1]
- Remove `#` and split the string in 3 substrings of 2 characters [1]
- Use the converter function from Task 3.1 [1]
- The output file is written to in correct format 16 rows of 13 columns with 3-list in each entry separated by comma [1]
- Correct values of rgb values entries in the list [1]

Task 3.3

- ask for input, function or otherwise [1]
- Uses loop to pass through input list to... [1]
  - ...convert the denary value to hexadecimal value via dictionary, array or otherwise [1]
- extracting quotient ... [1]
  - and remainder of the denary values [1]

- prepad `0` when denary value is between 0 and 9 [1]
- correct conversion of test denary values to hex values [1]
- Opens and closes the file for reading [1]
- Use the converter function defined in the same equation [1]
- Add `#` and join the 3 substrings into a string [1]
- The output file is written to in correct format 16 rows of 13 columns each entry separated by comma or space [1]
- Correct values of hex entries in the list [1]

Task 4.1

- generating random flip of 30 coins for one list [1]
- using for loop to generate 100 of such list [1]
- using `if`, or otherwise, to find number of lists that contain exactly 15 heads [1]
- number of elements in sample space, for probability calculation, is 100 [1]
- correct approximation obtained [1]

Task 4.2

- implement `myRandom` function correctly by using `random()` method or using high enough number to approximate the uniform distribution if using `randint()` method [1]
- `myRandom` replaced the uniform distribution function correctly [1]
- all 4 parameters p,n, c, h are asked as inputs [1]
- correct approximation obtained [1]

Task 4.3

- Using `for` loop to generate n lists that ... [1]
  - ... contain c random values which is either 0 or 1. [1]
- All 3 parameters p,n, c are asked as inputs [1]
- Procedure to find number of list that contains exactly h heads [1]
- (h, n_h) is a tuple stored within the list [1]

- correct number of h, which is between 0 and $c$inclusive [1]
- Reading the elements of the tuple correctly for printing. [1]
- print as required by the question [1]
- Evidence of testing [1]
- Histogram is accurate [1]