ENGG1811 Computing for Engineers (21T1)
Sample Final Exam

- Time allowed: 3 hours.

- This examination is worth 40% of the overall assessment for this course.

- In this exam, there are 5 questions and is worth 100 marks. Different questions have different marks.

- Each question requires you to submit a separate Python program file for marking. Note the following:

  - ❖ It is your responsibility to test your code thoroughly before submission. For some questions, an associated test file has been provided to help you to test your code.

  - ❖ Submission must be made using the provided web links (like assignment submissions).

  - ❖ Each question requires a specific file name and the submission system will only accept that particular filename.

  - ❖ Ensure that you **save** your file before submission. If the submission system accepts your file, it will run tests on your submitted file.

  - ❖ You can make multiple submissions during the exam. Only the last submitted file will be assessed.

Question 1 (20 marks)

It consists of Part a and Part b. You are **not** allowed to use numpy for this question.

Question 1a (10 marks)

Your task is to write a Python function `q1a_func` with the following `def` line:

```
def q1a_func(a_list,a,b):
```

where

- The input `a_list` is a Python list whose entries are of the type `int`. You can assume that `a_list` is non-empty.

- The inputs `a` and `b` are of the type `int`. You can assume that the value of `a` is less than that of `b`.

The function is required to **return** a value of the `bool` type.

The function `q1a_func` should return the `bool` value `True` if all the entries in `a_list` are greater than or equal to a and less than b; otherwise the function should return `False`. For example,

- If `a_list` is $[10, 5, 12]$, a is 5 and b is 13, then `q1a_func` should return `True`.

- If `a_list` is $[10, 5, 12, 11]$, a is 5 and b is 12, then `q1a_func` should return `False`.

- If `a_list` is $[10, 5, 12, 11]$, a is 6 and b is 13, then `q1a_func` should return `False`.

Requirements and testing:

- You must write the function `q1a_func` in a file with the filename `q1a.py`. The submission system will only accept this filename. A template file `q1a.py` has been provided.

- Your function must be able to work with any non-empty Python list whose entries are of type `int`.

- You can use the file `test_q1a.py` for testing.

- You do not need to submit `test_q1a.py`.

- Make sure that you **save** your file before submission.

Submission: You need to submit your solution file q1a.py using a WebCMS weblink, similar to your assignment submissions.

Question 1b (10 marks)

Your task is to write a Python function `q1b_func` with the following `def` line:

```
def q1b_func(a_list,m):
```

where

- The input `a_list` is a Python list of lists.

- The input `m` is of the type int.

The function is required to **return** a list.

We will use an example to explain what the function `q1b_func` should do. We will refer to the variable that `q1b_func` returns as `output`. The variable `output` is a list, which can be empty. In this example, we assume that `a_list` is the following list of lists:

```
[[3,4,8],[6,12],[7,8,14],[-1,-6,-9]]
```

If the maximum value of a list within `a_list` is less than or equal to `m`, then the index of that list should be included as an entry in `output`; otherwise the index of the list should not be included. The following examples show what the expected output should be for different values of `m`.

- If `m` is 14, then `output` should be the list `[0,1,2,3]` (or other lists with the same set of entries but ordered differently) because the maximum of each list in `a_list` is less than or equal to `m`

- If `m` is 12, then `output` should be the list `[0,1,3]` (order can vary).

- If `m` is 9, then `output` should be the list `[0,3]` or `[3,0]`.

- If `m` is 7, then `output` should be the list `[3]`.

- If `m` is -5, then `output` should be the list `[]`, which is the empty list.

Requirements and testing:

- You must write the function `q1b_func` in a file with the filename `q1b.py`. The submission system will only accept this filename. A template file `q1b.py` has been provided.

- Your function must be able to work with any non-empty Python list of lists.

- You can use the file `test_q1b.py` for testing.

- You do not need to submit `test_q1b.py`.

- Make sure that you save your file before submission.

Question 2 (20 marks)

For the data set "`data_sea_ice`" (used in the lab07 and lab09,) provide three *numpy* code segments for the following three tasks. You need to provide your answers in the file q2.py, at the required locations, and submit the file q2.py.

Notes:
- If required, you can add more *numpy* statements.
- You must **not** use a loop(s) for to answer the following three tasks.
- There are no test files for this question.

Task-1:   Calculate and save the average sea ice extent over the entire data collection in the variable **ans_task1** in the file q2.py (provided).

Task-2:   For each year, determine the number of half-months *that* exceeds the overall average calculated in the variable ans_task1 (during Task-1), and save your answer in the variable **ans_task2** in the file q2.py (provided).

Task-3:   Compute the mean sea ice extend in the first 6 months of years 2000-2009 (inclusive), and save your answer in the variable **ans_task3** in the file q2.py (provided).

Submission: You need to submit your solution file q2.py using a WebCMS weblink, like your assignment submissions.

Question 3 (20 marks)

If a person has a body mass m (measured in kilograms) and height h (measured in metres), then that person's body mass index (BMI) (measured in kg/m^2) is given by:

$$\text{BMI} = \frac{m}{h^2}.$$

The BMI is commonly used to assess whether a person's weight is healthy or not. The following table shows the classification of the health status according to the BMI.

BMI (kg/m^2)	Classification
Less than 18.5	Underweight
Greater than or equal to18.5 and less than 25	Healthy
Greater than or equal to 25 and less than 30	Overweight
Greater than or equal to 30	Obese

For example, a person with a BMI of 25 kg/m^2 is considered to be overweight.

Write a Python function that returns the classification of health status (one of the strings "Underweight", "Healthy", "Overweight" or "Obese"), given parameters m and h. The `def` line of the function should be:

```
def q3_func(m,h):
```

Requirements and testing:

- You must write the function `q3_func` in a file with the filename `q3.py`. The submission system will only accept this filename. A template file `q3.py` has been provided.

- You can assume that we will only use *positive* values of m and h for testing.

- You can use the file `test_q3.py` for testing.

- You do not need to submit `test_q3.py`.

- Make sure that you `save` your file before submission.

Submission: You need to submit your solution file q3.py using a WebCMS weblink, similar to your assignment submissions.

Question 4 (20 marks)

Note: The maximum mark that you can received for this question depends on whether you use loops to solve the problem or not. If you do **not** use loops, you can get the maximum; otherwise if you **do** use loops (either for or while), then the most you can get is 70% of the maximum.

Your task is to write a Python function `q4_func` with the following `def` line:

```
def q4_func(x,t):
```

where the first input `x` is a 1-dimensional numpy array and the second input `t` is a scalar of the type float. The function is required to compute and return a numpy array which has the same shape as `x`. In the following description, we will refer to the numpy array to be returned by the variable name `z`.

Let `x[i]` and `z[i]` be the element indexed by `i` in, respectively, the arrays `x` and `z`. The relationship between `x[i]` and `z[i]` is given by the following pseduo-code:

```
if  x[i] >  t
    z[i] = 1             # one
else if x[i] < -t
    z[i] = -1            # minus one
else
    z[i] =  x[i] / t
```

For examples:

- If `x[i]` is 10 and `t` is 5.2, then `z[i]` should take on the value of 1.

- If `x[i]` is -20 and `t` is 5.2, then `z[i]` should take on the value of -1.

- If `x[i]` is -2.6 and `t` is 5.2, then `z[i]` should take on the value of -0.5.

Requirements and testing:

- You must write the function `q4_func` in a file with the filename `q4.py`. The submission system will only accept this filename. A template file `q4.py` has been provided.

- Your function must be able to work with 1-dimensional numpy array `x` of any shape.

- You can assume we will always use positive `t` for testing.

- You can use the file `test_q4.py` for testing.

- You do not need to submit `test_q4.py`.

- Make sure that you `save` your file before submission.

Submission: You need to submit your solution file q4.py using a WebCMS weblink, similar to your assignment submissions.

Question 5 (20 marks)

Your task is to write a Python function `q5_func` with the following `def` line:

```
def q5_func(array,n):
```

where

- The input `array` is a 1-dimensional numpy array

- The input `n` is a positive integer. You can assume that the value of `n` is less than or equal to the number of entries in the array `array`.

The function is required to **return** a 1-dimensional numpy array with `n` elements. In the following, we will use `output` to denote the array that the function should return. We will now describe the relation between the elements of the arrays `array` and `output`.

For this description, we will assume that `array` has `m` entries where `m` is a positive integer bigger than or equal to `n`. Let `s` be the integer part of the quotient when `m` is divided by `n`. That is, you first divide `m` by `n`, and then discard the decimal part. For examples,

- If `m` is 14 and `n` is 4, then `s` is 3. This is because $\frac{14}{4} = 3.5$ and the decimal part of 0.5 is discarded.

- If `m` is 11 and `n` is 5, then `s` is 2. This is because $\frac{11}{5} = 2.2$ and the decimal part of 0.2 is discarded.

- If `m` is 10 and `n` is 2, then `s` is 5.

The relation between the elements of `array` and `output` are:

- `output[0]` is the sum of `array[0], array[n], ... array[(s-1)*n]`

- `output[1]` is the sum of `array[1], array[1+n], ... array[1+(s-1)*n]`

- ...

- `output[n-1]` is the sum of `array[n-1], array[n-1+n], ... array[n-1+(s-1)*n]`

The above relationships mean that each element of the array `output` is given by the sum of `s` elements in the array `array`.

There are four test cases in the file `test_q5.py` and we have explained how the output is computed from the inputs as comments in the file.

Requirements and testing:

- You must write the function `q5_func` in a file with the filename `q5.py`. The submission system will only accept this filename. A template file `q5.py` has been provided.

- Your function must be able to work with any 1-dimensional numpy array, and any value of positive integer `n`. You can assume that for all the tests, the value of `n` is no more than the number of elements in the array and the array is not empty.

- You can use the file `test_q5.py` for testing.

- You do not need to submit `test_q5.py`.

- Make sure that you save your file before submission.

Submission: You need to submit your solution file q5.py using a WebCMS weblink, similar to your assignment submissions.

------- End of the Exam ---------