ENGG1811 Computing for Engineers (21T1)
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)

You are **not** allowed to use numpy for this question.
Your task is to write a Python function `q1_funct` with the following def line:

```
def   q1_funct (a_list):
```

where

- The input `a_list` is a Python list whose entries are between -100 and 100 of type int. You can assume that `a_list` is non-empty.
- The function is required to return minimum positive value from a given list `a_list` (of type `int`). For this question, a value is a positive value if it is greater than zero. If there are no positive values in list `a_list`, the function should return zero.

For example,

- If `a_list` is `[7, 55, -10, 45, 2, -21]` then `q1_funct` should return `2`.
- If `a_list` is `[55, 18, 89, 55, 18, 89, 75, 92]` then `q1_funct` should return `18`.
- If `a_list` is `[-67, -56, -61, -12, -1]` then `q1_funct` should return `0`.
- If `a_list` is `[-81, -6, 0, -32, -31]` then `q1_funct` should return `0`.

Requirements and testing:

- You must write the function `q1_func` in a file with the filename `q1.py`. The submission system will only accept this filename. A template file `q1.py` has been provided.
- You can use the file test_q1.py for testing.
- You do not need to submit test_q1.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.
- Your answers will be tested on a modified data set (of the same size), so make sure your problem-solving approach is correct.

Please read the comments at the start of the file q2.py for more information.

Task-1 (4 marks): Calculate and save the number of sea ice extent over the entire data collection that is between two values **10** and **12** (inclusive) in the variable **ans_task1** in the file q2.py (provided).

Task-2 (8 marks): For each half month from the start of March to the end of July (inclusive), and for every year before 1985 or after 2010, calculate the total number of sea ice extent that is less than 12 and save your answer in the variable **ans_task2** in the file q2.py (provided).

Task-3 (8 marks): For every year between 1990 and 2000 (inclusive), calculate quarterly sea ice extent and arrange the years such that their corresponding third quarterly sea ice extents are sorted in ascending order. That is, the first year in the list is the year that has the lowest third quarterly sea ice extent, the second year has the second lowest third quarterly sea ice extent and so on. Save your answer in the variable **ans_task3** in the file q2.py (provided).

The expected answer for Task-3 is provided in the file q2.py.

For Task-3, first quarter includes months Jan, February, and March; second quarter includes months April, May, and June; third quarter includes months July, August, and September; fourth quarter includes months October, November, and December.

Requirements and testing:

- You must write your answers in the filename `q2.py`. The submission system will only accept this filename. A template file `q2.py` has been provided.
- There are no test files for this question.
- Your answers will be tested on a modified data set (of the same size), so make sure your problem-solving approach is correct.
- Make sure that you **save** your file before submission.

Question 3 (20 marks)

You have designed a chemical reactor with two chambers. The temperatures in the two chambers are `T1` and `T2` respectively.

If the *sum* of `T1` and `T2` is greater than 200 temperature units, then the reactor is in an alert state. The alert state is further divided into `alertHigh` and `alertLow`. The reactor is in `alertHigh` if either `T1` or `T2` is at or above 150 temperature units; otherwise it is in the `alertLow` state.

If the reactor is not in the alert state, then it is in the normal state. The normal state is further divided into two states `normalHigh` and `normalLow`. The reactor is in the `normalLow` state if both `T1` and `T2` are less than 50 temperature units, otherwise it is in the `normalHigh` state.

Your task is to write a Python function `q3_funct` with the following `def` line:

```
def   q3_funct (t1, t2):
```

where

- The input `t1` and `t2` are of type `double.`
- The function is required to return the state the reactor given the temperatures `t1` and `t2`. The function returns a string, which can be one of the following: "`alertHigh`", "`alertLow`", "`normalHigh`", "`normalLow`".

For example,

- If t1 is 300 and t2 is 220 then `q1_funct` should return "`alertHigh`".
- If t1 is 125 and t2 is 135 then `q1_funct` should return "`alertLow`".
- If t1 is 30 and t2 is 45 then `q1_funct` should return "`normalLow`".
- If t1 is 100 and t2 is 60 then `q1_funct` should return "`normalHigh`".

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 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.

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( array_a, b ) :
```

where the first input `array_a` is a 1-dimensional numpy array and the second input a is a scalar type float.

The function is expected to **return** one output (of type float).

We will use an example to illustrate what the function `q4_func` is required to do. For this example, `array_a` is `[-9.6, 0.9, 1.1, 2.8]` and the value of `b` is 1.0. The computations that `q4_func` has to do are:

a) For each number in `array_a` which is greater than or equal to `b`, you need to subtract `b` from the number and then divide the result by 2. For this example, the numbers 1.1 and 2.8 in `array_a` are greater than or equal to `b`. You need to compute (1.1 – b)/2 and (2.8-b)/2.

b) For each number in `array_a` which is less than `b`, you need to square it. For this example, numbers −9.6 and 0.9 in `array_a` are less than `b`. You need to compute $(-9.6)^2$ and $(0.9)^2$.

c) Add up the numbers that you have computed in Steps (a) and (b) above, and return the result as the output. For this example, you need to add (1.1 – b)/2, (2.8-b)/2, $(-9.6)^2$ and (0.9); you should get 93.92 which is the output that the function should return.

Note that the above example comes from test case 0 in the test file.

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.
- 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.

Question 5 (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 q5_func with the following `def` line:

```
def q5_func(expt_data, predicted_means):
```

where

- The input `expt_data` is a 2-dimensional numpy array,
- The input `predicted_means` is a 1-dimensional numpy array.

You can assume that the number of rows in `expt_data` is always the same as the number of elements in `predicted_means`, and neither of the arrays is empty.

The function `q5_func` is required to return an integer.

We will explain what the function should do using an example. In this example, `expt_data` is the following array:

```
[ [9.6, 3.5, 1.7, 2.8],
  [2.2, 1.1, 1.3, 1.6],
  [2.4, 1.0, 1.1, 1.3],
  [6.7, 6.9, 4.9, 0.7],
  [0.6, 2.0, 0.3, 0.5] ]
```

You can interpret each row of the array as the measurements obtained from an experiment. Since the shape of `expt_data` is 5-by-4, there are 5 experiments, and each experiment has 4 measurements. You can use the given `expt_data` to compute the mean of the measurements for each experiment. We will refer to these means as the experimental means. For this example, the experimental means are:

```
[4.4, 1.55, 1.45, 4.8, 0.85]
```

where 4.4 is the mean of 9.6, 3.5, 1.7 and 2.8; 1.55 is the mean of 2.2, 1.1, 1.3 and 1.6; and so on.

The predicted means of the experiments are stored in the given 1-dimensional array `predicted_means`, which in this example is:

```
[4.3, 1.6, 1.7, 1.1, 2.7]
```

If we compute the absolute difference of the corresponding elements of the experimental means and `predicted_means`, we have:

```
[0.1, 0.05, 0.25, 3.7, 1.85]
```

where `0.1 = |4.3 - 4.4|, 0.05 = |1.6 - 1.55|` etc.

The largest absolute difference occurs at the array index 3 (value of 3.7), and the function `q5_func` should return this index.

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.

- 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.

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