MATH36031 Problem solving by computer.
Project 1 - deadline 28th October 2022, time 1100hrs. Submission of the project is via Blackboard.

1. **The Catalan constant** The Catalan constant $\mathcal{C}$ defined as

$$\mathcal{C} = \sum_{0}^{\infty} \frac{(-1)^k}{(2k+1)^2} = \frac{1}{1^2} - \frac{1}{3^2} + \frac{1}{5^2} + \cdots \approx 0.915965594177219$$

is a mathematical constant named after the French and Belgian mathematician Eugène Charles Catalan. Despite its appearance in many seemingly unrelated integrals and functions, basic questions like whether it is irrational or not are still open. In many situations, it is more convenient to approximate complicated constants like this by a rational approximation.

You are asked to find the best such approximations p/q such that $|\frac{p}{q} - \mathcal{C}|$ is smallest among all possible positive integers p and q such that $p+q$ is smaller than or equal to a given positive integer.

Task A: Write a function `AppCat` such that `[p,q] = AppCat(N)` returns the best pair of integers p and q, such that $|\frac{p}{q} - \mathcal{C}|$ is smallest amongst all positive integers and $p+q$ is less than or equal to N. Record your result for $N = 2022$. Note: you can just copy the above value of the constant with fifteen decimal points, otherwise you have to call the symbolic constant and convert it into double precision via:

```
catconst = double(catalan)
```

The first few lines of your code should look like:

```
function [p , q ] = Appcat ( N )
%APPCat Approximates the catalan constant by the rational number p/q
% APPCat Approximates the catalan constant by the rational number
% p /q , among all pairs of positive integers (p , q ) such that p +q <= N


catconst =0.915965594177219
```

Notes: (a) you can assume that N is a positive integer greater than one and there is no need to check the validity of the input; (b) if there are more than one pair of numbers (for instance, if for the input integer N, both pairs $[p_1, q_1]$ and $[2p_1, 2q_1]$ give the best approximation), return the pair with the smallest $p+q$ value.

2. **Cubic taxicab numbers** A *cubic taxicab number* is a positive integer that can be expressed as the sum of two positive cube numbers in two or more distinct ways, for example

$$1729 = 1^3 + 12^3 = 9^3 + 10^3.$$

Task B: Write a function `CubicTaxicabNum` such that `[a,b,c,d,M] = CubicTaxicabNum(N)` returns the smallest cubic taxicab number $M = a^3 + b^3 = c^3 + d^3$ say greater than or equal to N, and the four values a, b, c and d. (You can assume the input N is a positive integer). Record your output for $N = 36031$.

```
function [a,b,c,d,M] = CubicTaxicabNum ( N )
%CUBICTAXICABNUM returns the smallest cubic taxicab number M
% CUBICTAXICABNUM returns the smallest cubic taxicab number
% M=a^3+b^3=c^3+d^3 greater than
% or equal to N
%
```

Task C: Suppose now that we define T_n to be the smallest integer that can be expressed as the sum of two positive integers cubed in n distinct ways. As an example

$$T_1 = 2 = 1^3 + 1^3, \quad T_2 = 1729 = 1^3 + 12^3 = 9^3 + 10^3.$$

Use or modify your codes for Task B to find T_3 outputting the corresponding pairs of integers. You may assume that $87000000 < T_3 < 88000000$ to reduce the computing time in finding T_3.

Additional Information

- All coding must be done in MATLAB and you are required to submit your MATLAB functions and M-files via the Blackboard submission box. Project reports **in pdf form only** should be submitted via the Turnitin submission box. Remember the Turnitin software will automatically scan reports for plagiarism.
- Please ask if you need help on any commands, or whether there are built-in commands/functions to accomplish certain tasks (especially important if you think you have a good approach to the questions, but do not know the related commands).
- The default datatype is double (decimal number), and be aware of possible side effects when using the numbers as integers. Remember that the same question can be solved by different approaches, and the same approach can be implemented in different ways. All relevant commands should be covered during the lectures or tutorial exercises, but you are free to explore your own. Make critical judgement to choose the best approach/implementation.
- Aim for efficiency of the code, which is an additional marking criteria, besides the generic rubric. Although you only need to record the answer for the given input, make sure that the computational time or memory does not increase significantly with larger input parameters (these issues will be mentioned constantly during the class demonstrations).
- List the complete code of the whole function at the end of each question, or in an appendix. Make your source code more readable, by keeping the indentation and stylistic features, and can be copied from the electronic file.
- The results reported in your report **must** be reproducible from your codes. Remember that markers will be able to run the codes in case of any doubts and any inconsistencies between reported results and actual results from running codes will lead to reports being marked down.

Guidelines for the report.

1. Have a look at the generic rubric and frequently asked questions, which is given in Blackboard in the Projects folder and about how your report will be marked. The rubric also describes the intended learning outcomes about what you are expected to achieve at the end.

2. Avoid copying (too many) sentences directly from the project description, and try to restate the problem with your own words or examples if possible.

3. You may use your report in the future as evidences of written work, so take it seriously.

4. Your target audience is a fellow student on your course: explain the questions so that the report can be understood without this project description and your approach can be implemented in another computer. The report should indicate to the reader how well you understand the problem and the approach you have taken, the validation and other checks that you made to ensure your results are credible. Reports submitted containing codes only and with no explanations of how the problem was solved, will result in a failing mark, even though the codes may work perfectly well and give the correct answers.

5. Balance the explanation of the approach and the comments in the code. Avoid under-commenting and over-commenting.

6. Aim for precision and clarity of writing (discussed in Week 5).

7. Keep your page length **not exceeding eight A4 pages,** with a font size no smaller than 11, and page margins no smaller than 2cm. There is no need for a title page for a relative short report like this. **If more than 8 pages are submitted only the first 8 pages will be marked and the rest of the submission will be ignored.**

8. Since there is no final exam, you are advised to spend at least 15 hours on each project.

9. The submission box (via Blackboard and Turnitin) for each project will be open two weeks before the deadline, and you are encouraged to submit an early draft to see how Turnitin works on Blackboard. Only your last submission will be marked, and DO NOT submit anything after the deadline. Any late penalty will be applied by the Teaching and Learning Support Office according to the Undergraduate Student Handbook, and any extension has to be approved from the Office too (not from the lecturer).