IT1007 Practical Exam 2018

During PE

- Do NOT close your coursemology browser.
  - If you accidentally close it, you need the invigilator to re-open it for you. Resulting in losing time
- Put your ID on the top of your table
- Only 5 submissions per question
  - We will NOT grant you extra test runs or "finalize submission" if you accidentally used them up.
- Copy ONLY the required functions
  - Do NOT submit your test cases/test code
- Because we have numerical answers, Coursemology *may* tell you that you are "wrong" because your output numbers are not exactly the same with the sample answer. You should just submit with confidence.
- Click "Finalize Submission" to submit your code. You will not be able to amend your code after you have finalized your submission.
- You can access the help/reference from IDLE (F1).
- You are not allowed to import extra packages other than the ones that are already imported in our skeleton code (except for Question 4).
- You must name your functions exactly as the questions stated.
- Submit your code 15 min before the end
  - At the last 10 min of the PE, the network will be ***jammed heavily,*** and it will take a long time to submit your code (> 15 min).
  - Any unsubmitted code will deem as zero mark. It's your own responsibility to submit your code correctly.

Grading

We will also grade your programs according to your style and performance. Namely, in order to gain full marks, you should not have bad programming styles and slow computations, such as

- Unnecessary lengthy code
- Forbidden functions, such as using extra packages. **If you import forbidden function, it will result in ZERO mark**.
- But you can assume that the inputs of your function will be in the right type. For example, we will not call your function as "`matchResistors('abc',10)`" in Question 3.

Tips

- Strategize!!!!!!!! Don't' get stuck on ONE question
  - Some questions are harder and with lower marks, and some are easier but with higher marks. Read through ALL the questions first and organize the order of questions to work with.

Question 1 [20 marks]: Computing Cosine Function

In this question, you cannot import any packages or functions except the one and only one "`pi`" from the math package. Remember to copy the line "`from math import pi`" into couresmology in your submission.

If the angle x is in ***radian***, you can compute the cosine function by Taylor series:

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = \frac{1}{1} - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} + \cdots$$

Of course, we cannot compute the perfect cosine until n equals to infinity (and no one can). Let's say we compute the first $i + 1$ terms of Taylor series such that,

$$\text{myCos}(x) \simeq \sum_{n=0}^{i} \frac{(-1)^n x^{2n}}{(2n)!}$$

Write a function '`myCos(angle,k)`' to take in two arguments, the angle in *degree* and the number of terms k, and it will *return* the sum of the first k terms in the Taylor series. (Note that $i = k - 1$)

For example, if you use a calculator and you can get the answer of cos(45) to be 0.7071067811865475… And your function should be able to produce approximated answers shown in the right sample runs.

For full mark, your function should be able to handle a large number of terms, e.g. $k > 100$.

```
>>> print(myCos(0,5))
1.0
>>> print(myCos(45,1))
1.0
>>> print(myCos(45,2))
0.6915748624659576
>>> print(myCos(45,3))
0.707429206709773
>>> print(myCos(45,10))
0.7071067811865475
>>> print(myCos(60,1))
1.0
>>> print(myCos(60,80))
0.5000000000000001
>>> print(myCos(90,80))
-9.607321133294986e-18
```

Question 2: Smallest Four Number [25 marks]

In this question, you cannot create any list in your function, except that you can create one list and one tuple of four elements each to store the results. Also, you cannot modify the original list, and remember that list is NOT passed by values. Your function should work if the input is a tuple also. And you cannot use the built-in function sort(). And you cannot import any packages or functions.

Write a function, '`findSmallest4(L)`', which takes in a sequence L and returns a tuple of the smallest four numbers in the sequence. You can assume the input list contains only integers, and its length is at least 4. Your function should return a tuple of four integers from the largest to the smallest.

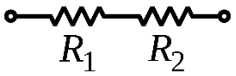
Sample output:

```
>>> print(findSmallest4([5, 3, 2, 6 ,7 ,8 ,1, 99]))
(5, 3, 2, 1)
>>> print(findSmallest4((1, 2, 3, 4, 5, 6, 8, 9, 1)))
(3, 2, 1, 1)
```

Question 3: Finding Resistor Pairs [30 marks]

We are going to fly a lot of drones to scout out an area!

Everything is ready except that they need some resistors in their main circuits! For each drone, it needs two extra resistors that must be connected in series to produce the desire total resistance. Namely, total resistance $R = R_1 + R_2$.



For example, if you need a total resistance of 10 ohms, you can connect two resistors with resistances 2 ohms and 8 ohms, or another pair of 3 ohms and 7 ohms.

You are given as many drones as possible, and they all need the same resistance R. In your hands, you are given a lot of resistors such that every resistor is unique and none of them has the same resistance with each other. And of course, every resistor has a positive non-zero value of resistance. Your task is, given a list L that contains the resistance values of all the resistors, find out all the pairs of resistors that can produce a total sum of R ohms. For example, if you have a list of resistors:

```
resistorList = (75,80,90,77,88,91,60,74,73,70,55,93,59)
```

You can call your function to find out what are the combinations for different total resistance. These two runs find all the resistors that can sum up to 150 ohms and 152 ohms.

```
>>> print(matchResistors(resistorList,150))
[(59, 91), (60, 90), (70, 80), (73, 77)]
>>> print(matchResistors(resistorList,152))
[(59, 93), (75, 77)]
```

Write the function '`matchResistors(L,R)`' to *return* a list of tuples. And each of the tuple contains a pair of resistor in the list L that sum up to R. Here is a sample run of the function:

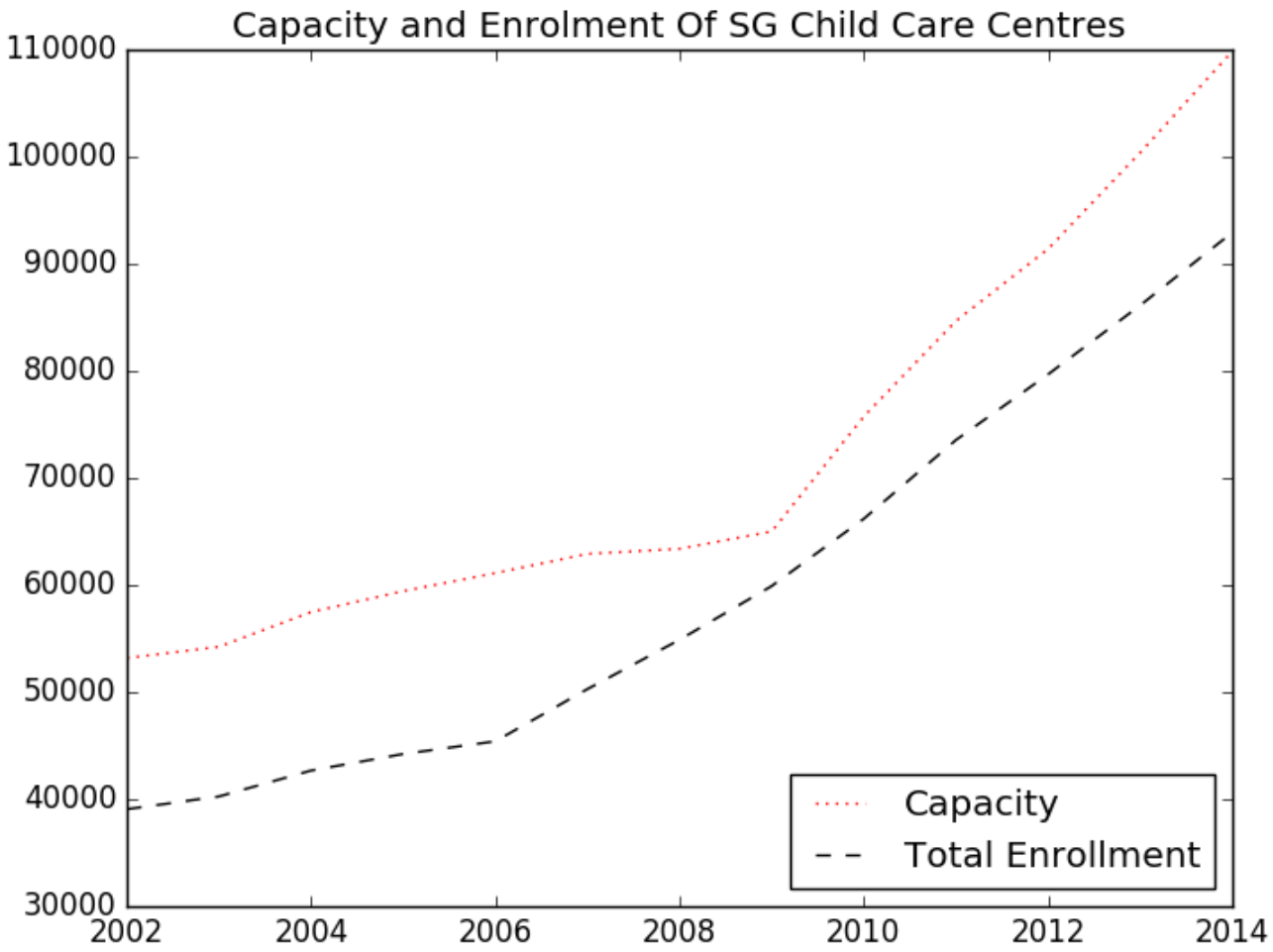
Again, here are some rules for your function in order to gain full marks:

- You cannot modify the original input. And your function should work with inputs of both lists and tuples. You cannot hardcode the input
- You cannot import any packages or functions but you may use the built-in function `sort()`.
- Every item in your output must be *unique*. Namely, you cannot output the same pair twice, even if you swap the two values.
- Your list must be sorted by the smaller value of each tuple in ascending order. And each pair must be sorted in ascending order also.
- You cannot use any Set operations

Question 4: Plotting Childcare Centres Capacities and Enrollments [25 marks]

In this question, you can import any packages you want. But remember to copy the import lines in your code also. And there is no test run for this question in coursemology.

You are given a data file "**`child-care-centres-capacity-and-enrolment-annual.csv`**" from data.gov.sg. Your job is to plot and display the following graph from that data file.



Write the function "`plot_child_care()`" to display the graph above. (And you do not need to save any graph.) Put the title and the legend at the same position as shown above. The line for "Capacity" is a red dotted line. And the line for "Total Enrollment" is black dash line.

- End of Paper -