COMPS258

Computer Programming and Problem Solving

Autumn 2021 Presentation

Assignment 3

Please submit this Assignment by 23rd March 2022

Preamble

I hope you have learned a lot in this course so far. We have now come to the part covering more challenging topics, including data structures, algorithms and problem solving. These topics are fundamental to programming and experienced programmers will often employ knowledge on data structures, algorithms, and problem solving in their work. I would encourage you to press on and pay attention to the programming aspects of the topics.

Please spend enough time on this Assignment and start as early as possible. You will probably find this Assignment most challenging of all so please be prepared for the challenges. This Assignment covers Units 1 to 6, and mainly Units 3, 4, 5 and 6.

Writing and calling functions is an important technique. Python provides many built-in and third-party functions that enable programmers to do things quicker and easier. It is a good practice to research the Internet and look for the functions that are useful. You should try reading the reference materials such as API and know how to search for them. For example, you can try 'Python string split' or 'Python split tutorial' on Google to look up information about the split function for Python strings.

You are at a disadvantage if you only stick to the functions covered in the study materials.

For this assignment, you are especially suggested to attempt Q2(c). The question looks long but it is along the content in the study unit. Q2(b) and Q3(a) can be difficult and suitable for students who aim for excellent grades.

I think we all have our burden in our family and work. Hopefully we can stand well against the different challenges. We can make up our mental attitude and how we live our life. Let me quote from Winston Churchill (the UK wartime prime minister).

Success is going from failure to failure without a loss of enthusiam.

Let us not lose our enthusiasm and our hope.

General Requirements

Between 20% to 40% of a question is attributed to the general requirements. *The following lists the specifics of the general requirements and their relative importance for this Assignment*	
Specifics	
Programming Style	Your program solutions should follow a good programming and formatting style. Marks are awarded/deducted for good/poor programming and formatting style.
Program Design	Overly inefficient code.

Question 1 Programming (52%)

Specific Requirements of Question

Specifics	
Data Validation	The **range** of the input from the user and parameter values should be checked to avoid causing your program to crash. This is about checking the range and see if the value making sense and not causing an execution error. For example, the height of square should be check against negative value (which does not make sense) (which would cause the program to crash). However, you need not deal with the cases that have been assumed in the question itself.

(a) Complete the following function that generates and return a list of random integers in the range 1 to 49 (both inclusive). The function should accept a parameter **`N`** indicating how many random integers to be included in the list. The function should assert that the parameter **`N`** is greater than 0. Assume that **`N`** is always an integer.

```
def genRanList(N):
```

Submit **q1a.py**. [8 marks]

(b) Write a program to break down a given amount of money into the standard denominations of $1 or more. Here the standard denominations include $500, $100, $50, $20, $10, $5, $2, $1. Well, nobody likes $1000 bank notes. The list of broken down money should have minimal number of bank notes and coins.

The program should accepts integers that is $5 or more. The program first asks for the amount to break down. If the input is valid, the break down list is printed, and if otherwise, an error message is printed. The program then asks if the user wants to enter another amount, and repeat if the answer is positive. Refer to the following to an example execution.

```
Enter the amount you wish to break down: 800
Here's your change:
$ 500 : 1
$ 100 : 3
Want to enter another amount (y/n): y
Enter the amount you wish to break down: 278
Here's your change:
$ 100 : 2
$ 50 : 1
$ 20 : 1
$ 5 : 1
$ 2 : 1
$ 1 : 1
Want to enter another amount (y/n): y
Enter the amount you wish to break down: 3649
Here's your change:
$ 500 : 7
$ 100 : 1
$ 20 : 2
$ 5 : 1
$ 2 : 2
Want to enter another amount (y/n): y
Enter the amount you wish to break down: 7.50
Please enter a valid dollar amount (whole numbers only).
Want to enter another amount (y/n): y
Enter the amount you wish to break down: 2
Amount cannot be less than $5.
Want to enter another amount (y/n): n
Bye and see you next time.
```

Submit **q1b.py**. [8 marks]

(c) To expel the pandemic in the new year, Master Seven has suggested having a lucky password. A lucky password should also be secured. She suggests a lucky password should satisfies all the following conditions.

- Between 8 and 32 characters
- Contains only uppercase, lowercase, digits, and the punctuations `'+'`, `'-'`, `'*'`, `'/'`.
- Contains at least one digit and only one of the punctuation characters.
- Contains the substring '77' in the password at least once.

Complete the following function that accepts a password (as a string) as a parameter and returns True if it is a lucky password, and False if otherwise. Assume that the input parameter contains a string.

```
    def isLuckySeven(pwd):
```

Submit **q1c.py**. [12 marks]

(d) A file **EPLTeams.csv** is given to you that contains the number of games played and the number of winning games for some soccer teams.

Every line in the file contains a comma separated list of 3 items: soccer team name, number of games played and number of winning games, in this order. The winning percentage is the number of winning games divided by number of games played. The first line in the file should be ignored

Write a complete program that analyzes the file and finds out the team with the highest winning percentage. If there are two or more teams with the same highest, choose any.

The program should print the findings to the screen on a single line, and then print `'Analysis finished'` on the next line before the end of execution. The program has no input from user.

The program should also print `'IO Error'` if there is an error in file operations.

Submit **q1d.py**. [8 marks]

(e) This question is about the BigO notation.

(i) Estimate the time taken for every step in a procedure is 0.001 second. There are 4 different algorithms for implementing the procedure, and the number of steps for each algorithm is described in the following Big-O notation. Assume that N is 1000. The first answer is given to you.

Big-O of Implementation of the Procedure	Time Taken (s)
O(N)	1.000
O(lg N)	
O(N lg N)	
O(N^2)	
O(N^3)	

(ii) There are two algorithms: Algorithm A takes 500 seconds to complete the processing of data and the time taken is found the same regardless of the amount of data. Algorithm B takes 1 second to process 100 data and 4 seconds to process 200 data. Which algorithm is more likely to have performance of O(1)? Give reasons.

Submit your answers in **q1e.txt**. [8 marks]

(f) Apply **selection sort** to the following words by hand according to the following requirements.

- Ascending order (*Largest* at the right-most).
- Use left-to-right progress order (the sorted area expands from the left to right).
- When two words are compared, the "larger" word is the longer one, and if the length is equal, the alphabetically order is used. The case is ignored in the comparison (e.g. 'A' and 'a' are considered the same).

```
Parrot Sparrow Duck Pigeon Owl Gull Eagle Peacock
```

Submit your work in **q1f.txt**. Show your working step by step on each line in the text file. On the last line, report the total number of comparisons required. [8 marks]

Question 2 Middle Level Programming (36%)

Specific Requirements of Question

Specifics	
Error Tolerance	The functions you write should handle erroneous input and parameters and report an error if necessary.
Efficiency	You should avoid overly inefficient coding in your programs.

(a) The following is a summation series of 4 terms (N = 4).

$$\frac{1}{1^2}+\frac{1}{3^2}+\frac{1}{5^2}+\frac{1}{7^2}$$

Do you know that if we add more terms, the result of the summation is getting closer and closer to π^2 divided by 8. Therefore we can find π from this equation, with N is the number of terms in the summation series.

$$\frac{1}{1^2}+\frac{1}{3^2}+\frac{1}{5^2}+\frac{1}{7^2}+\cdots+\frac{1}{(N\times 2-1)^2}=\frac{\pi^2}{8}$$

$$\pi^2=\frac{8}{1^2}+\frac{8}{3^2}+\frac{8}{5^2}+\frac{8}{7^2}+\cdots+\frac{8}{(N\times 2-1)^2}$$

Turn the above into a recursive function. Complete the following recursive function that returns the value of π^2 given a certain N as the parameter.

```
def find_pi_square(N):
```

Assume that N is a positive integer <= 500. Submit **q2a.py**. [8 marks]

(b) The Nth Fibonacci number is defined as below (for N >= 1)

$$Fib(N)=Fib(N-1)+Fib(N-2)$$
$$Fib(1)=Fib(2)=1$$

The first and the second Fibonacci numbers are both 1.

An implementation of finding the Nth Fibonacci number is given to you as the function **`Fib`** in the skeleton program q1bskeleton.py.

(i) Add a main program to print a series of Fibonacci numbers from A to B, where A and B are obtained from the input. The following shows an example execution.

```
Enter the start of the series (A): 5
Enter the end of the series (B): 8
5 5
6 8
7 13
8 21
```

(ii) [Challenging] Note one thing Fib(8) is found by that Fib(8) = Fib(7) + Fib(6), therefore requiring the result of Fib(6). However, Fib(7) = Fib(6) + Fib(5), meaning that Fib(6) was already calculated once for Fib(7). Therefore in the above example, Fib(6) was calculated three times.

The first time calculating Fib(6) is unavoidable. However, the second time and more seems a waste of effort. If the result of the first time calculation is stored away, when the second time Fib(6) is needed, the result can be looked-up instead of calculation again.

Assume that N is a positive integer <= 1000.

Make such a change to the function **`Fib`**.

Submit **q2b.py**. [12 marks]

(c) The skeleton program file q2c.py contains an implementation of QuickSort as covered in Activity 6.16 in the function named **`quickSort`**. The program sorts a list of number in ascending order.

(i) Implement the function **`quickSortNew`** that has a second optional parameter **`reverse`** for specifying whether the sorting is in ascending order (the default that **`reverse`** is **`False`**) or descender order (**`reverse`** is **`True`**). The other functions such as **`quickSortHelper`** and **`partition`**, have been modified with the optional parameters. Make change to the function **`partition`** to handle both ascending order and descending order.

Hint: the following four lines in the function are responsible for partitioning the list into two halves with the left smaller than the right. This is suitable for ascending order. Consider what it should be like for descending order.

```
while left <= right and aList[left] <= pivot_val:
    left += 1
while right >= left and aList[right] >= pivot_val:
    right -= 1
```

(ii) The first number is used as the pivot in the current implementation. Modify the program to adopt the randomized pivot strategy to select the pivot value. Refer to Unit 6 for a more detailed discussion of this enhancement.

(iii) Compare the running times of the regular quicksort from Unit 6 and the new version based on the randomized pivot strategy. You should implement the code in the program file **q2c_compare.py** and using timeit (number=1) to carry out tests based on two benchmark input.

Test 1: A randomized ordered sequence of floating-point numbers (between 0 to 10000) with input sizes 100, 200, 400, and 800. Fill in the following table in the file q2c.txt.

Randomly ordered	**100**	**200**	**400**	**800**
quickSort				
Randomized quickSort				

Test 2: A reverse sorted ordered sequence of floating-point numbers (between 0 to 10000) with input sizes 100, 200, 400, and 800. Fill in the following table in the file q2c.txt.

Reverse sorted ordered	**100**	**200**	**400**	**800**
quickSort				
Randomized quickSort				

(iv) Write down which version performed better in q2c.txt. Explain why.

Submit all your solutions in **q2c.py, q2c_compare.py** and **q2c.txt**. [16 marks]

Question 3 Challenging Programming (12%)

Specific Requirements of Question

Specifics	
Error Tolerance	The functions you write should handle erroneous input and parameters and report an error if necessary.
Efficiency	You should avoid overly inefficient coding in your programs.

(a) Write a complete program that prints out the list of files in the current directory, along with their file sizes and last modified datetime stamps. Any sub-directories are ignored. At the end, print the total file size.

It accepts two optional command-line arguments:

1	Sort field (single sort key only)	“name” for filename, “size” for file size, “date” for last modified datetime stamp, default is “name”.
2	Sort order	“asc” for ascending, “desc” for descending, default is “asc”.

A sample execution of the program with both command line arguments provided is shown below.

```
$ python q3a.py date desc
Filename        Size    Last Modified
fileSorter.py / 1461 / Sat Jan 30 01:40:48 2021
optimized_fib.py / 273 / Fri Jan 29 10:53:45 2021
tic-tac-toe-bot-master.zip / 37180 / Thu Jan 28 15:12:03 2021
tictactoe.py / 8157 / Thu Jan 28 14:59:45 2021
getComputerMove.py / 1131 / Thu Jan 28 14:55:22 2021
Total file size: 48203
```

Research how you access the file system using the functions in the **`os`** module. You may use Python’s built-in **`sort()`** function to perform the sort.

Submit **q3a.py**. [12 marks]

Appendix A Submission Summary

The following table shows the files that may be submitted. The submitted files will be computer-processed. Failure to use the correct file names may earn you zero mark.

Filename	***Format***	***Description***
q1a.py	Python Source	Question 1(a)
q1b.py	Python Source	Question 1(b)
q1c.py	Python Source	Question 1(c)
q1d.py	Python Source	Question 1(d)
q1e.txt	Text File	Question 1(e)
q1f.txt	Text File	Question 1(f)
q2a.py	Python Source	Question 2(a)
q2b.py	Python Source	Question 2(b)
q2c.py	Python Source	Question 2(c)
q2c_compare.py	Python Source	Question 2(c)
q2c.txt	Text File	Question 2(c)
q3a.py	Python Source	Question 3(a)

Appendix B Extension Application

Please submit the assignments on or before the cut-off dates.

Your tutor is authorized to grant extension of up to seven days.

Only in the circumstances of sickness (with medical proof), long business trips (at least a week long and happen within 2 weeks of the cut-off dates), and exceptional events would extension of 8 to 21 days be considered. We will reply your extension application through the OLE Assignment Records and Extension System.

No marks will be awarded for any late assignments without prior approval obtained from the Course Coordinator and/or the Dean.

If you are in an exceptionally difficult situation and unable to submit within 21 days, you may make an application to the Dean of School. In such situations, please email us at alui@hkmu.edu.hk and we will tell you what to do.

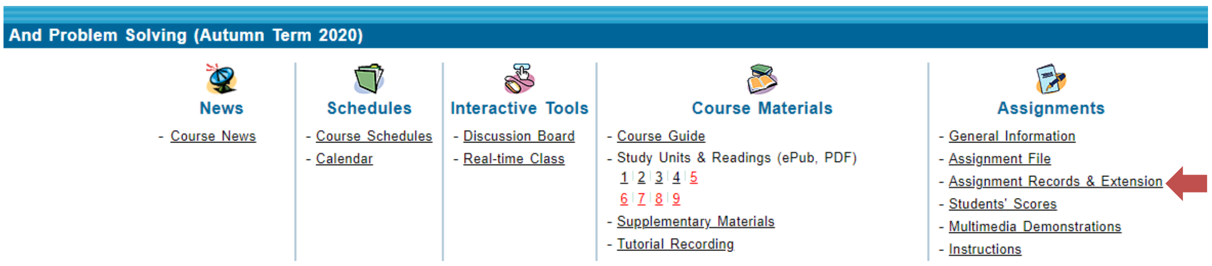
Appendix C Electronic Submission

Assignment submission is a three-step process.

Step 1. Start working on your assignment as early as possible, so you can submit your work by the cut-off date. Follow the file names specified in the assignment questions. Place all the questions in a folder.

Step 2. Package your work (files) into a ZIP file. ZIP files are a common way in the Internet era for packaging and sending multiple files. You will gather all your work files into one ZIP file using a package such as WinZip. Name your ZIP file with your student ID followed by the suffix .zip (for example, 01234567.zip). This is an additional measure to avoid mistakes.

Step 3. Submit the ZIP file at the OLE (Online Learning Environment) , Records & Extension System (https://ole.hkmu.edu.hk)


And Problem Solving (Autumn Term 2020)
News
- Course News
Schedules
- Course Schedules
- Calendar
Interactive Tools
- Discussion Board
- Real-time Class
Course Materials
- Course Guide
- Study Units & Readings (ePub, PDF)
1 2 3 4 5
6 7 8 9
- Supplementary Materials
- Tutorial Recording
Assignments
- General Information
- Assignment File
- Assignment Records & Extension
- Students' Scores
- Multimedia Demonstrations
- Instructions