

COMPS258

Computer Programming and Problem Solving

Autumn 2021 Presentation

Assignment 4

Please submit this Assignment by 4th May 2022

Preamble

We wish that you will enjoy good health and live a good life. The fact that you are studying this course externalizes your desire to improve yourself. I hope by now you have learned something useful about programming. In addition, the later part of this course should provide you with some useful basic understanding about concepts for advanced programming.

This Assignment covers Units 5 to 8, especially 6 to 8. Some questions are quite challenging. You should not expect to complete all questions, especially if you have already obtained the scores to pass the OCAS. Just try what you can do.

Please note that no late submission will be allowed. The university has a policy of restricting granting extension for the last assignment. The reason is that the tutors would have a reasonable period to mark the assignment before the examination comes. Only under some exceptional circumstances would a few days be granted.

Important: Make sure that you report any score discrepancy. You can check your Assignment scores in the OLE or Student Intranet. If you make a Assignment 4 submission but the score does not appear in the OLE 21 days after the due-date, please contact your tutor. After that, if your score still does not appear in the OLE 28 days after the due-date, please email us at alui@hkmu.edu.hk. Changing the score after this date will be very difficult for us. Please help us by checking your score.

General Requirements

<i>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</i>	
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 (56%)**Specific Requirements**

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

- (a) You are given **q1a.py** that contains three mystery functions, namely `funcA`, `funcB`, and `funcC`, that process a Python list.
- Evaluate the scalability of the three functions empirically, based on the time taken to process lists of sizes 500, 1000, 2000 and 4000. Refer to Activity 6.19 for how to use the `timeit` third party module. [12 marks]

- Add code to use `timeit` for the time taken and complete the program **q1a.py**. The program should be executed once to generate the time taken for all the cases in the table below that summarizes the time taken for processing lists of various sizes by the three functions. The time measurements should be in seconds down to 4 decimal places. Fill in the results in **q1a.txt**.

List Size	500	1000	2000	4000
FuncA				
FuncB				
FuncC				

- From the time measurements, write down the performance of `funcA` and `funcB` in Big-O notation. Put your answer in **q1a.txt**.
- Refer to **Tables 6.17 and 6.18**. From the time measurements of `funcC`, compare the performance of `funcC` (better or worse) to Linear time and Quadratic time? Suggest the performance of `funcC` in Big-O notation. Put your answer in **q1a.txt**.
- Rank the three functions from best scalability to worst scalability. Explain your answer. Put your answer in **q1a.txt**.

Submit **q1a.py** and **q1a.txt**. [12 marks]

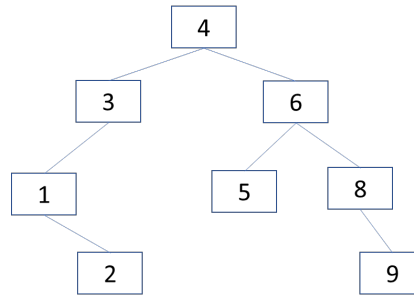
- (b) A palindrome is a string that reads the same when reversed. Usually, the spaces are ignored, and the case of the letter is irrelevant to determine if a string is a palindrome. Study Activity 7.6 and the section it belongs to. The function `isPalindrome` is provided for you in the skeleton program file **q1b.py**. Modify the function according to the following

- All the digits and punctuations and other non-letter characters should be ignored
- Consecutive letters (both upper and lower cases) should be considered as a single letter. For example, 'Aa', 'aA', 'aa', 'aaaa', 'AAAAA', are all considered as 'a'.

For example, the input of 'a67a' is a palindrome because the non-letter characters are ignored.

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

(c)	<p>Write a complete Python program that reads in data from a CSV file about soccer teams in England top level league, and then prints out the following findings.</p> <ul style="list-style-type: none"> • The name and the points of top 5 teams with the highest points. • The number of teams with name containing "United" and won fewer than 500 games. • The number of teams that have played over 2000 games and scored over 5000 goals. <p>The CSV file (EnglishLeagueAllTime.csv) contains lines of data, and each line (except the first line which is a header) contains a comma-separated list of the past performance of the soccer teams. The items of every team on each line are, in this order, the team name (Team), the number of games played (Play), the number of games Won, Drawn or Lost (Win, Draw, and Lose respectively), the number of goals scored and conceded (For and Against respectively), the goal difference (Diff) and the number of points (Points)</p> <p>The program should first ask for the name of the CSV file. Then the program should print the findings with suitable output decoration. Assume that the format of the lines of data is correct. IO exception handling is needed.</p> <p>Submit q1c.py. [12 marks]</p>
(d)	<p>Your friend Doris has been studying LinkedList and she is using the file linkedList.py in Activity 7.5. One day she has written a new function for the LinkedList class. The function is called insertMystery. She is challenging you whether you know what the function is doing. All you know is that the function adds data to the linked list.</p> <p>To test her function, a main program is provided for you. Part of that is shown below.</p> <pre> if __name__ == '__main__': # create the linked list with a single node containing 'Anders' aList = SLinkedList(Node("Anders")) # insert nodes using insertMystery aList.insertMystery("Betsy", "") aList.insertMystery("Davy", "") aList.insertMystery("Martha", "") aList.insertMystery("Jodie", "") aList.insertMystery("Katie", "") aList.traverse() </pre> <p>Answer the following questions and put the answers in q1d.txt.</p> <ol style="list-style-type: none"> Explain the purpose of the function insertMystery, in particular the parameter after. Explain the stopping condition of the loop in the insertMystery function. Compare the coding of the function append and the function insertMystery. Highlight the major difference and explain the code. Write down the time complexity of the function insertMystery. Explain your answer. <p>Submit q1d.txt. [8 marks]</p>
(e)	<p>Consider the following binary search tree (BST) and answer the following questions.</p>

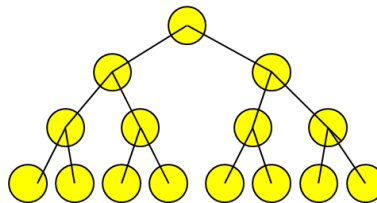


- (i) List the nodes visited in in-order traversal
- (ii) List the nodes visited in pre-order traversal
- (iii) List the nodes visited in post-order traversal
- (iv) Give the height of the tree
- (v) Explain how the node should be replaced if the node 1 is deleted
- (vi) Explain how the node should be replaced if the node 4 is deleted

For node deletion, use the methods (including replacement with immediate successor and others) described in Unit 7. Use the template file and submit your answer in **q1e.txt**. [8 marks]

- (f) A perfect binary tree looks like the following. It is a full binary tree, in which all nodes have two children except the leaf nodes, and all leaf nodes are at the same level.

Perfect Binary Tree



Study the `binarytree` module from Unit 7. The module allows the building and manipulation of different types of binary trees. Use the module to write a complete Python program containing the parts in the same order as the questions.

- (i) Use 7 carefully chosen integers to build a perfect binary search tree with height of 2 (the above example has height of 3). Refer to Unit 7 on how to build a tree with specific numbers. You should use the `Node` class of the `binarytree` module. Then print the tree (as in top the Page 54 in Unit 7).
- (ii) The `binarytree` module offers a way to build perfect binary tree using the `bst` function (<https://www.geeksforgeeks.org/binarytree-module-in-python/>). Use the function to build a perfect binary search tree with height of 3. Then print the tree
- (iii) For the tree built in part (ii) above,
 - Print the total number of nodes
 - Print the total number of leaves
 - Print the left most node in the tree. Hint: use in-order traversal.
 - Print the right most node in the tree. Hint: use in-order traversal.

Submit **q1f.py**. [8 marks]

Question 2 Data Analysis Mini-Project with Python (24%)**Specific Requirements of Question**

Specifics	
Efficiency	You should avoid overly inefficient coding in your programs.

- (a) In this question you will take up the challenge of handling the raw Covid-19 infection case data from CHP Hong Kong. You should first study Unit 8 for this question. You will also find it useful to research the web and look for methods of how to do certain things with Python Pandas and Matplotlib. [24 marks]

Tasks 1 to 12 are the basic tasks and they worth 16 marks. The last task is open ended, and it is worth 8 marks.

Prepare the dataset of confirmed Covid-19 infections in Hong Kong for further analysis. You may download the latest version of the file (in English) from the CHP at:

http://www.chp.gov.hk/files/misc/enhanced_sur_covid_19_eng.csv

The CSV file stopped updated since early February 2022. You may use the one provided in the skeleton files instead.

To help you understand the fields in the data set, please read pages 1 and 2 of the data dictionary at https://www.chp.gov.hk/files/pdf/nid_spec_en.pdf.

Below are the first few rows of the dataset used here (version used is from 20-Jan-2022).

	Case no.	Report date	Date of onset	Gender	Age	Name of hospital admitted	Hospitalised/Discharged/Deceased	HK/Non-HK resident	Classification*	Case status*
0	1	23/01/2020	21/01/2020	M	39.0	NaN	Discharged	Non-HK resident	Imported case	Confirmed
1	2	23/01/2020	18/01/2020	M	56.0	NaN	Discharged	HK resident	Imported case	Confirmed
2	3	24/01/2020	20/01/2020	F	62.0	NaN	Discharged	Non-HK resident	Imported case	Confirmed
3	4	24/01/2020	23/01/2020	F	62.0	NaN	Discharged	Non-HK resident	Imported case	Confirmed
4	5	24/01/2020	23/01/2020	M	63.0	NaN	Discharged	Non-HK resident	Imported case	Confirmed

This exercise demonstrates the use of Python to perform an analysis of the dataset and hopefully it will result in some interesting findings.

Perform the following steps in a Jupyter notebook called q2a.ipynb:

1. Load all columns in the data file `enhanced_sur_covid_19_eng.csv` into a pandas dataframe.
2. The column 'Name of hospital admitted' contains no data. Tidy up the dataframe by removing the column using the drop function.

```
cases_df.drop(columns=['Name of hospital admitted'])
```

3. Let us check the status of all the cases and whether they are HK residents or not. The following code uses the groupby function call to count the frequencies of all categories in the "Hospitalised/Discharged/Deceased" column and then plot a pie chart. Complete the code also for the "HK/Non-HK resident" column.

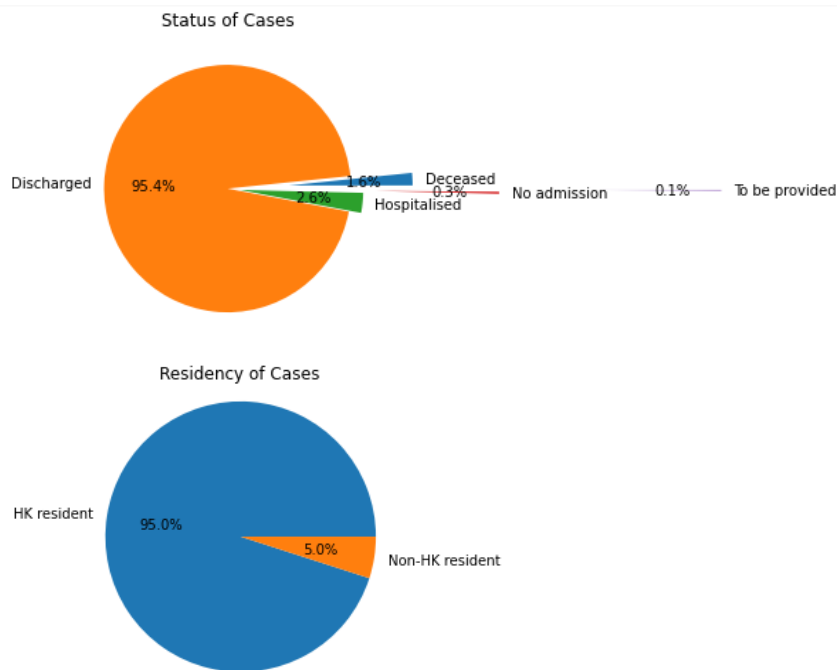
```

st_stat = cases_df.groupby(['Hospitalised/Discharged/Deceased']).count()
st_stat = st_stat[:, 'Case no. ']
print(st_stat)

fig, ax = plt.subplots(2, figsize=(15, 12))
ax[0].pie(st_stat, labels=st_stat.index, autopct='%1.1f%%', explode=(0.5, 0, 0.1, 1.2, 3.0))
ax[0].set_title('Status of Cases')

# ADD YOUR CODE to plot to ax[1]

```



4. The "Date of onset" column indicates the date of COVID symptom first appearing for the case. Some cases are so-called asymptomatic, and the word "Asymptomatic" is there instead of the date. It is more convenient for data analysis if there is a column for "Asymptomatic".
- The apply method runs the function **setAsymptomatic** on each row of the dataset (axis=1). You can find out if the case is asymptomatic by looking at the 'Date of onset' column.
- Use the following template.

```

def setAsymptomatic(row):
    pass
    # ADD YOUR CODE HERE
cases_df['Asymptomatic'] = cases_df.apply(setAsymptomatic, axis=1)
cases_df.tail()

```

	Case no.	Report date	Date of onset	Gender	Age	Name of hospital admitted	Hospitalised/Discharged/Deceased	HK/Non-HK resident	Classification*	Case status*	Asymptomatic
13096	13097	20/01/2022	Asymptomatic	M	11.0	NaN	To be provided	HK resident	Epidemiologically linked with imported case	Asymptomatic	Y
13097	13098	20/01/2022	18/01/2022	M	14.0	NaN	To be provided	HK resident	Epidemiologically linked with imported case	Confirmed	N
13098	13099	20/01/2022	18/01/2022	M	16.0	NaN	To be provided	HK resident	Epidemiologically linked with imported case	Confirmed	N
13099	13100	20/01/2022	18/01/2022	F	1.0	NaN	To be provided	HK resident	Epidemiologically linked with local case	Confirmed	N
13100	13101	20/01/2022	Asymptomatic	M	12.0	NaN	To be provided	HK resident	Epidemiologically linked with imported case	Asymptomatic	Y

5. The columns "Date of onset" and "Report date" are useful for calculation of the risk of spreading the virus in the community. For ease of computation, the non-date cells in the "Date of onset" column that contains strings (such as "Asymptomatic") should be replaced with **None**. In addition, both columns should be changed to **datetime** type.

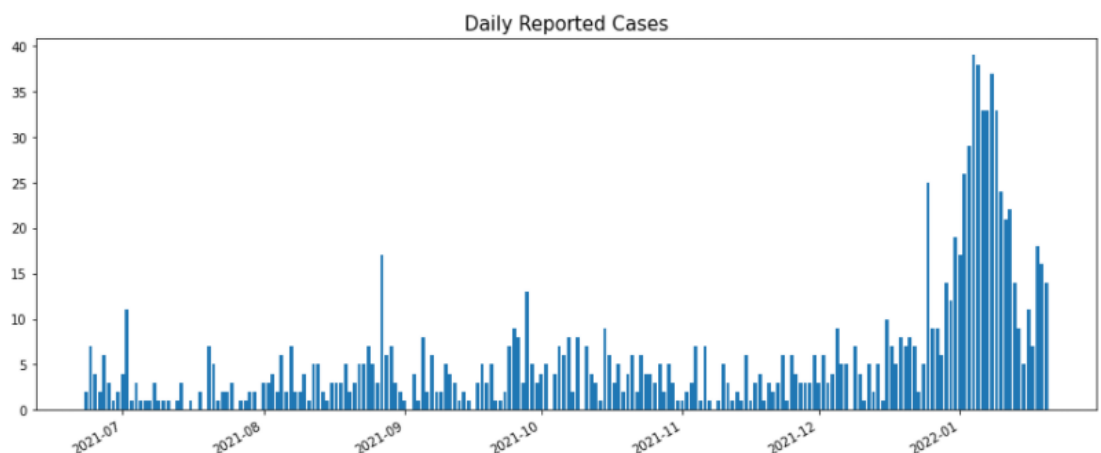
The following show the code for converting the "Date of onset" column. Use the code for this task and also write a similar statement for converting "Report date".

Note: Cases 503, 1104, 1369, 1370, 1373 contains non-date value in "Date of onset" column.

```
cases_df['Date of onset'] = pd.to_datetime(cases_df['Date of onset'],
infer_datetime_format=True, errors='coerce')
# ADD CODE FOR 'Report date'
```

6. Let work out the trend of COVID-19 cases since the start of the pandemic. The groupby function is again used to count the number of cases for each day. The date value of the 'Report date' column is used as the key. The commented-out statement is for selecting the last period for the plotting.

```
freq_stat = cases_df.groupby(cases_df['Report date'].dt.date).count()
# freq_stat = freq_stat[-200:]
fig, ax = plt.subplots(1,figsize=(15, 6))
fig.autofmt_xdate()
ax.bar(freq_stat.index, freq_stat['Case no.'])
ax.set_title('Daily Reported Cases', fontsize=15)
```



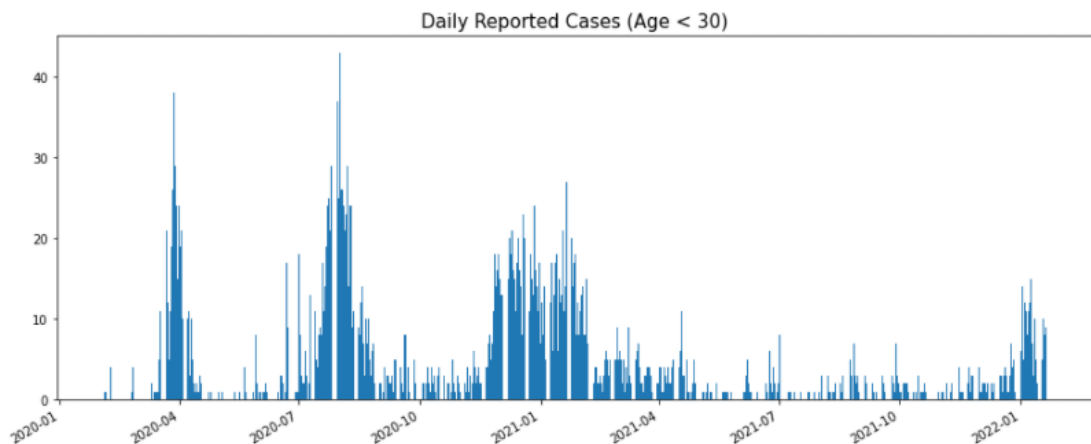
7. It is often insightful to cut out a part of the dataset for analysis. In the following code, the female cases are selected using the dataframe selector. The variable `sp_cases` contains a new dataframe that consists of female cases only.

```
sp_cases = cases_df[cases_df['Gender'] == 'F']
```

Follow the example above.

- Select the cases of age less than 30 years old
- Plot the daily reported cases of age less than 30 years old as in the last task. Note that the cut out rows are now in the `sp_cases` dataframe.

You should see something similar to the following plot.



8. For a long while, there were no local cases. Only imported cases were detected. The information containing this is found in the "Classification*" column. The following shows example of selecting the cases out and then plot them.

```
local_cases = cases_df[(cases_df['Classification*'] == 'Local case') |
(cases_df['Classification*'].str.contains('Local case', case=False))]

freq_stat = local_cases.groupby(local_cases['Report date'].dt.date).count()

fig, ax = plt.subplots(1, figsize=(15, 6))
fig.autofmt_xdate()
ax.bar(freq_stat.index, freq_stat['Case no.'])
ax.set_title('Daily Reported Cases (Local)', fontsize=15)
```

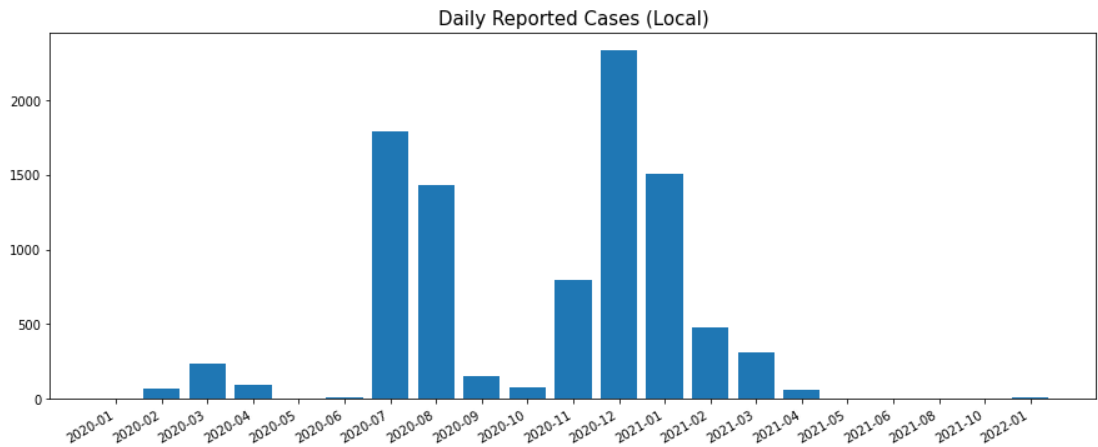
The local cases are marked as 'Local case' and 'Epidemiologically linked with local case'. However, another label 'Epidemiologically linked with imported case' should also be considered as local cases. Modify the example accordingly.

9. It is useful to look at the monthly trend in addition to the daily trend. One convenient way to do that is to create a new column with a monthly label. The following example will convert dates into labels such as '2020-01', '2021-12', etc.

```
def set_monthly(row):
    return str(row['Report date'])[0:7]
```

```
cases_df['Monthly'] = cases_df.apply(set_monthly, axis=1)
print(cases_df['Monthly'])
```

10. Plot the monthly trend of local cases (similar to Task 8). Replace the column used for the **groupby** with the new 'Monthly' column.



11. The difference between the Report date and Date of onset seems important for assessing of risk of a case. A new column 'Lapse Days' can be calculated by subtracting 'Date of onset' from 'Report date'. However, we must know that some cells in the 'Date of onset' and 'Report date' columns contain None. Use the following example to create the new column.

```
def cal_lapse(row):
    if (row['Asymptomatic'] != 'Y'):
        try:
            report_date = pd.to_datetime(row['Report date']).date()
            onset_date = pd.to_datetime(row['Date of onset']).date()
            lapse = report_date - onset_date
            return lapse.days
        except:
            return None
    else:
        return None

cases_df['Lapse Days'] = cases_df.apply(cal_lapse, axis=1)
cases_df[0:81]
```

12. Save the prepared dataset as "ready_cases.csv". Then print out some statistics that may be useful to know. Examples of printing the statistics are given below.

```
# ADD YOUR CODE FOR SAVING the dataframe to CSV

select_df = cases_df[cases_df['Hospitalised/Discharged/Deceased'] == 'Deceased']
print('Mean Age of Deceased: ', select_df['Age'].mean())
print('Mean Age of Deceased by Gender: ',
      select_df.groupby(['Gender'])['Age'].mean())
```

```
print('Mean Lapse Days: ', cases_df['Lapse Days'].mean())
```

[Challenging Task]

13. This task is open ended. Based in the above prepared dataset of Covid-19 cases in Hong Kong, carry out an analysis to reveal some useful information. Your analysis should satisfy the following technical requirements.

- Add this task to the end of the same Jupyter Notebook (q2a.ipynb) and begin the new section with markdown 'Challenging Task'.
- The score for this part depends on the quality of analysis, the level of techniques used, and the explanation of the analysis and findings.
- You should explain your analysis as markdown in the notebook so that the user can follow.
- You should include one or more plots using matplotlib.
- The maximum number of words in the markdown is 150.

Some suggestions of additional data analysis.

- Investigate the daily trend of new cases in relation to other factors
- Investigate the population who were more vulnerable to infection
- Investigate any trend between the reporting date and onset date

Make sure the cell(s) relevant to a task are labeled with the task number for identification.

Submit your Jupyter notebook **q2a.ipynb**, any datasets used (e.g., the original case CSV files), and the output file **ready_cases.csv**.

Question 3 Mini Programming Project (20%)**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) This is a mini-programming project that allows you to try using what you have learned in this course to deal with a larger programming problem. It is alright not to complete the whole project. Just try your best to do as much as possible, and mark down the progress as required in the program header.

Write a program that schedules the meetings in a conference room (“A”) such that as many meetings as possible can be held on a given day.

Any meetings that cannot be held in conference room A can be moved to another conference room (“B”) if it is unoccupied during the requested period, otherwise the meeting is rejected.

The meeting requests are stored in a CSV file `meeting_requests.csv`. A sample is shown below.

```
Request,Start Time,End Time
1,1:30 PM,2:30 PM
2,8:00 AM,9:00 AM
3,9:30 AM,11:00 AM
4,11:00 AM,12:00 PM
5,3:00 PM,5:00 PM
6,8:30 AM,9:30 AM
7,3:00 PM, 5:30 PM
8,10:00 AM, 11:00 AM
9,11:00 AM,11:30 AM
10,2:30 PM,3:00 PM
```

The occupancy schedule in Conference Room B is stored in another CSV file `conf_B_sched.csv`. A sample is shown below.

```
Start Time, End Time
9:30 AM, 10:30 AM
2:00 PM, 4:00 PM
```

The output for the sample inputs above are:

```
Conference A Schedule: 2, 3, 9, 1, 10, 5
Conference B Schedule: 6, 4
Rejected Meetings: 8, 7
```

You may adapt the algorithm outlined in [page 56 to 58 of Unit 5](#) (for the basic interval scheduling problem) or another suitable algorithm of your choice.

Additional requirements:

- The program does not interact with the user. It reads data from the files, calculates and terminates after printing the results to the screen.
- The two conference rooms are available for booking only from 08:00 AM to 07:00 PM. Meeting requests with start or end time outside of the allowed period should be rejected.

Submit **q3a.py**. [20 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.

<i>Filename</i>	<i>Format</i>	<i>Description</i>
q1a.py	Python Source	Question 1(a)
q1a.txt	Text File	Question 1(a)
q1b.py	Python Source	Question 1(b)
q1c.py	Python Source	Question 1(c)
q1d.txt	Text File	Question 1(d)
q1e.txt	Text File	Question 1(e)
q1f.py	Python Source	Question 1(f)
q2a.ipynb	Jupyter Notebook	Question 2(a)
ready_cases.csv	CSV Text File	Question 2(a)
?.csv	Other CSV Text File	Question 2(a)
q3a.py	Python Source	Question 3(a)

Appendix B Extension Application

Please note that no late submission will be allowed. The university has a policy of restricting granting extension for the last assignment. The reason is that the tutors would have a reasonable period to mark the assignment before the examination comes. Only under some exceptional circumstances would a few days be granted.

If you are in an exceptionally difficult situation and unable to submit at the cut-off date, please email me at alui@hkmu.edu.hk and then we will discuss about what you may 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 <ul style="list-style-type: none">- Course News	 Schedules <ul style="list-style-type: none">- Course Schedules- Calendar	 Interactive Tools <ul style="list-style-type: none">- Discussion Board- Real-time Class	 Course Materials <ul style="list-style-type: none">- Course Guide- Study Units & Readings (ePub, PDF)<ul style="list-style-type: none">1 2 3 4 56 7 8 9- Supplementary Materials- Tutorial Recording	 Assignments <ul style="list-style-type: none">- General Information- Assignment File- Assignment Records & Extension- Students' Scores- Multimedia Demonstrations- Instructions