Question 1

```
# Task 1.1

import pymongo, json
client = pymongo.MongoClient("127.0.0.1", 27017)
db = client.get_database("Travel")
coll = db.get_collection('Flight')

with open ('TICKET.json') as file:
    data = json.load(file)
client['Travel']['Flight'].insert_many(data)

result = coll.find()
for doc in result:
    print(doc)

client.close()
```

Output:

```
{'_id': ObjectId('6274f0ea145173440470ff19'), 'country':
'Australia', 'city': 'Sydney', 'price': 904, 'company': 'B Airways',
'durationHour': 8}
{'_id': ObjectId('6274f0ea145173440470ff1a'), 'country':
'Australia', 'city': 'Melbourne', 'price': 1628, 'company': 'Q
Airlines', 'durationHour': 7}
{'_id': ObjectId('6274f0ea145173440470ff1b'), 'country':
'Australia', 'city': 'Perth', 'price': 881, 'company': 'S Airlines',
'durationHour': 5}
{'_id': ObjectId('6274f0ea145173440470ff1c'), 'country': 'UK',
'city': 'London', 'price': 1373, 'company': 'L Airlines',
'durationHour': 19, 'stop': 2}
{'_id': ObjectId('6274f0ea145173440470ff1d'), 'country': 'UK',
'city': 'London', 'price': 1561, 'company': 'S Airlines',
'durationHour': 17, 'stop': 1}
{'_id': ObjectId('6274f0ea145173440470ff1e'), 'country': 'UK',
'city': 'Manchester', 'price': 1708, 'company': 'E Air',
'durationHour': 25, 'stop': 1}
{'_id': ObjectId('6274f0ea145173440470ff1f'), 'country': 'UK',
'city': 'Manchester', 'price': 2917, 'company': 'B Airways',
'durationHour': 17, 'stop': 1}
{'_id': ObjectId('6274f0ea145173440470ff20'), 'country': 'Japan',
'city': 'Osaka', 'price': 821, 'company': 'S Airlines',
'durationHour': 10, 'stop': 1}
{'_id': ObjectId('6274f0ea145173440470ff21'), 'country': 'Japan',
'city': 'Tokyo', 'price': 1028, 'company': 'J Airlines',
'durationHour': 7}
{'_id': ObjectId('6274f0ea145173440470ff22'), 'country': 'Japan',
'city': 'Tokyo', 'price': 1124, 'company': 'Air J', 'durationHour':
7}
```

```
# Task 1.2
import pymongo

# connect to the database
client = pymongo.MongoClient("127.0.0.1", 27017)
db = client.get_database("Travel")
coll = db.get_collection('Flight')

# update flight without stops
query = {"stop":{"$exists":False}}
update = {"$set":{"stop":0}}
coll.update_many(query, update)

# list all information of flights by company S Airlines
print("Flights by S Airlines:")
result = coll.find({"company":"S Airlines"}, {"_id":0})
for doc in result:
    print(doc)
print()

# list city and price for flights under 10 hours and under $1500
print("Flights under 10 hours and under $1500")
query={'$and':[{'durationHour':{'$lt':10}}, {'price':{'$lt':1500}}]}
result = coll.find(query)
print("city \t price")
for doc in result:
    print(doc['city'] + '\t', doc['price'])
print()

# flight information of the cheapest ticket
print('Cheapest Ticket')
print(coll.find().sort("price")[0])
print()


client.close()
```

```
Flights by S Airlines:
{'country': 'Australia', 'city': 'Perth', 'price': 881, 'company':
'S Airlines', 'durationHour': 5, 'stop': 0}
{'country': 'UK', 'city': 'London', 'price': 1561, 'company': 'S
Airlines', 'durationHour': 17, 'stop': 1}
{'country': 'Japan', 'city': 'Osaka', 'price': 821, 'company': 'S
Airlines', 'durationHour': 10, 'stop': 1}

Flights under 10 hours and under $1500
city    price
Sydney  904
Perth   881
Tokyo   1028
Tokyo   1124

Cheapest Ticket
{'_id': ObjectId('6274f0ea145173440470ff20'), 'country': 'Japan',
'city': 'Osaka', 'price': 821, 'company': 'S Airlines',
'durationHour': 10, 'stop': 1}
```

**Marker's Comment:*

*For questions to display all information, we can simply print the document. Table formatting is not required since NoSQL has no fixed schema.*

*Arrange the codes for each sub task in one cell.*

Task 2

```
# Task 2.1
def task2_1(filename):
    task1 = open(filename, 'r')
    word_list = []
    word = ""

    text = task1.read()
    for char in text:
        if char.isalpha():
            word = word + char.lower()
        elif len(word) > 0:
            word_list.append(word)
            word = ""
        else:
            pass
    if len(word)>0:
        word_list.append(word)

    task1.close()
    return word_list

print(task2_1('LYRICS.txt'))

# Task 2.1 marker's comments:
Many students did not close the file or comment that file
automatically closes if we use "with open".
The edge case "zip - unzip it" gave many students
problems, with many getting [zip,-,unzip,it] or
['zip','','unzip','it'] and a total output of 889.
```

Output:

```
882
```

```
# task 2.2
def task2_2(list_of_words):
  unique_words = []
  for index in range(1, len(list_of_words)):
    if (list_of_words[index] == 'it') and \
    (list_of_words[index-1] not in unique_words):
      unique_words.append(list_of_words[index-1])

    max_length = 0
    for i in unique_words:
      if len(i) > max_length:
        max_length = len(i)
    sorted_words = []
    for i in range(1, max_length + 1):
      each_length_words = []
      for word in unique_words:
        if len(word) == i:
          each_length_words.append(word)
      each_length_words.sort()
      sorted_words = sorted_words + each_length_words
  return sorted_words

result = task2_2(task2_1('LYRICS.txt'))
print(result)
print(len(result))

# Task 2.2 marker's comments:
If the question did not ask for a specific sorting
algorithm, sorted() and list.sort() are allowed.
While list.sort(key=len) can be used in A-levels, students
are advised to think of solutions using first principles.
```

Output:

```
['buy', 'cut', 'fix', 'pay', 'rip', 'use', 'burn', 'call',
'code', 'drag', 'drop', 'fill', 'find', 'load', 'lock',
'name', 'play', 'plug', 'read', 'save', 'scan', 'send',
'snap', 'surf', 'tune', 'turn', 'view', 'work', 'zoom',
'break', 'bring', 'check', 'click', 'crack', 'cross',
'erase', 'leave', 'paste', 'pause', 'point', 'press',
'print', 'touch', 'trash', 'unzip', 'watch', 'write',
'change', 'charge', 'format', 'rename', 'scroll', 'unlock',
'update', 'rewrite', 'upgrade']
56
```

```
# task 2.3
def task2_3(list_of_words, number):
    from random import randint
    word_length = 0
     return_string = ""

    for i in range(number):
        rand = randint(0, len(list_of_words) - 1)
        return_string += list_of_words[rand]
        list_of_words.remove(list_of_words[rand])

        if i != number - 1:
            return_string += ' it, '
        else:
            return_string += ' it.'
    return return_string

print(task2_3(result, 8))
# Task 2.3 marker's comments:
Random.randint() is the only random method that is
required. If you know of other methods in the package,
please ensure that you can satisfy the requirements in the
question.
Eg. Sample() (or even shuffle()) are good easy methods to
use for this question, but choice() is choosing with
replacement, which means it is non-distinct.
```

Task 2 Alternatives:

```python
# task 2.1
def task2_1(filename):
    with open(filename, 'r') as task1:
        word_list = []
        word = ""

        text = task1.read().lower()
        text = ''.join([char for char in text if
char.isalpha() or char in [" ","\n"]])
        word_list = text.strip().split()

    # auto close
    return word_list

print(task2_1('LYRICS.txt'))
print(len(task2_1('LYRICS.txt')))
```

```python
# task 2.2
def task2_2(list_of_words):
    unique_words = []
    for curr in list_of_words:
        if curr == 'it' and prev not in unique_words:
            unique_words.append(prev)
        prev = curr

    unique_words.sort()
    unique_words.sort(key = len)
    # can do this since list.sort() is a stable sort

    return unique_words

result = task2_2(task2_1('LYRICS.txt'))
print(result)
print(len(result))
```

```python
# task 2.3
def task2_3(list_of_words, number):
    from random import randint, shuffle, sample
    # not choice (which choose with replacement)

    lyrics = sample(list_of_words,number)
    return_string = " it, ".join(lyrics) + " it."

    return return_string

print(task2_3(result, 8))
```

3. Name your Jupyter Notebook as `TASK3_<your name>_<ct>.ipynb`

Solution:
Task 3.1

```
#Task3.1
class Node:
    def __init__(self, data, leftP=-1, rightP=-1):
        self._data = data
        self._leftPtr = leftP
        self._rightPtr = rightP

    def getData(self):
        return self._data

    def getLeftPtr(self):
        return self._leftPtr

    def getRightPtr(self):
        return self._rightPtr

    def setData(self, data):
        self._data = data

    def setLeftPtr(self, leftP):
        self._leftPtr=leftP

    def setRightPtr(self, rightP):
        self._rightPtr=rightP

    def __str__(self):
        return f"data:{self._data},left:{self._leftPtr},
right:{self._rightPtr}"
```

```
#Task3.1

class BSTree:
    ##initialise()
    def __init__(self):
        self._thisTree = [None] * 10
        self._root = -1
        self._nextFree = 0
        for i in range(9):
            self._thisTree[i]=Node("", i+1, -1)
        self._thisTree[9]=Node("", -1, -1)
    ##add()
    def add(self, newItem):
        if self._nextFree!=-1:  ##not full
            free = self._nextFree
            self._nextFree = self._thisTree[free].getLeftPtr()

            if self._root==-1:  #empty tree, set node as root
                self._thisTree[free].setData(newItem)  ##set newItem
                self._thisTree[free].setLeftPtr(-1)  ##set Left
                self._thisTree[free].setRightPtr(-1) ## set right
                self._root = free
            else:
                added= False
                probe = self._thisTree[self._root]  #root
                while added == False:
                    if probe.getData()<newItem:  #go to right node
                        if probe.getRightPtr() != -1:
                            probe=self._thisTree[probe.getRightPtr()]
                        else:
                            probe.setRightPtr(free)
                            self._thisTree[free].setData(newItem)  ##set
newItem
                            self._thisTree[free].setLeftPtr(-1)  ##set Left
                            self._thisTree[free].setRightPtr(-1) ## set right
                            added=True
                    else:
                        if probe.getLeftPtr()!= -1:
                            probe=self._thisTree[probe.getLeftPtr()]
                           #go to left node
                        else:
                            probe.setLeftPtr(free)
                            self._thisTree[free].setData(newItem)  ##set
newItem
                            self._thisTree[free].setLeftPtr(-1)  ##set Left
                            self._thisTree[free].setRightPtr(-1) ## set right
                            added=True
        else:
            print("Tree is full")
```

```
    def _displayInOrder(self, root):
        if root!=-1:
            self._displayInOrder(self._thisTree[root].getLeftPtr())
            print(self._thisTree[root].getData())
            self._displayInOrder(self._thisTree[root].getRightPtr())

    def displayInOrder(self):
        self._displayInOrder(self._root)
```

```
    def _displayPostOrder(self, root):
        if root!=-1:
            self._displayPostOrder(self._thisTree[root].getLeftPtr())
            self._displayPostOrder(self._thisTree[root].getRightPtr())
            print(self._thisTree[root].getData())

    def displayPostOrder(self):
        self._displayPostOrder(self._root)
    def display(self):
        print(f"root:{self._root}")
        print(f"nextFree:{self._nextFree}")
        for i in range(10):
            print(f"index:{i}, data:{self._data},
                  left:{self._leftPtr},right:{self._rightPtr}")
            ## alternative
            ##print(f"index:{i}, {self._thisTree[i]}")
```

Task 3.2

```
#Task 3.2
#Initialise
bst = BSTree()
filename="NAMES.txt"
#open and close file.
f = open(filename, 'r')
lines = f.readlines()   #read all lines in the file
f.close()

#process each line
for line in lines:
    word=line.strip("\n")
    bst.add(word)  #add into BSTree object


#call display
print("call display:")
bst.display()
print()
##call displayInorder
print("call displayInorder:")
bst.displayInOrder()
print()
print("call displayPostorder:")
bst.displayPostOrder()
```

```
Tree is full
call displayTree:
root:0
nextFree:-1
index:0, data:Geo, left:1, right:2
index:1, data:Don, left:5, right:8
index:2, data:Tim, left:4, right:3
index:3, data:Vin, left:-1, right:6
index:4, data:Leo, left:7, right:9
index:5, data:Ace, left:-1, right:-1
index:6, data:Wyn, left:-1, right:-1
index:7, data:Kai, left:-1, right:-1
index:8, data:Ela, left:-1, right:-1
index:9, data:Roy, left:-1, right:-1
```

```
call displayInorder:
Ace
Don
Ela
Geo
Kai
Leo
Roy
Tim
Vin
Wyn

call displayPostorder:
Ace
Ela
Don
Kai
Roy
Leo
Wyn
Vin
Tim
Geo
```

**Marker's Comment:*

*Many students failed to initialize the free space list correctly. It should contain a list of "empty nodes". Many failed to use the created "empty node" from the free space list when a new item is to be added to the tree.*

*Some students miss the specification in which the add method is coded using.*

*All outputs are to be shown clearly in the submitted file.*

```
# Task 4
def check(guess, answer):
# both are strings
    code = ''
    for i in range(4):
        # correct guess gives green
        if guess[i] == answer[i]:
            code += 'G'
        # wrong but still within the number # gives yellow
        elif guess[i] in answer:
            code += 'Y'
        # not even in the number gives red
        else:
            code += 'R'
    return code

import socket
import random

mysocket = socket.socket()
mysocket.bind(('127.0.0.1',12345))
mysocket.listen()
newsocket, addr = mysocket.accept()

newsocket.sendall(b'GUESS\n')

tries = 0
answer = str(random.randint(1000, 9999))

while True:
    data = b''
    while b'\n' not in data:
        data += newsocket.recv(1024)

    tries += 1
    guess = data.decode().strip()
    code = check(guess, answer)

    if code == 'GGGG':
        newsocket.sendall(b'WIN\n')
        break
    elif tries == 5:
        newsocket.sendall(b'LOSE\n')
        newsocket.sendall(answer.encode()+b'\n')
        break
    else:
        code += '\n'
        newsocket.sendall(code.encode())
        newsocket.sendall(b'GUESS\n')

newsocket.close()
mysocket.close()
```

*Marker’s Comment:

- A few students did not submit a Python file. **Note that in A level marking, zero mark will be given if the files submitted are not formatted as required.** For this exam, only one mark is deducted. Always read questions and follow the instructions closely.
- Many students are not familiar with the syntax of socket methods. For example, bind expects the input to be a tuple of two elements, not two input parameters
- For random integers, randint(a, b) includes b but randrange(a, b) excludes b
- Many students did not implement the protocol of `\n`
- The client file given tells information about the messages in the transmission and we shall design our server program accordingly