Q1

```
# socket server
import socket
server = socket.socket()
server.bind(('127.0.0.1',12345))
server.listen()
s, addr = server.accept()

with open('WORDS.txt', 'r') as f:
    wordList = []
    for line in f:
        wordList.append(line.strip())

from random import randint
size = len(wordList)
word = wordList[randint(1, size) - 1]

for i in range(5):
    letter = s.recv(1024).decode()
    if letter in word:
        s.sendall(b'YES')
    else:
        s.sendall(b'NO')

guess = s.recv(1024).decode()
if guess == word:
    s.sendall(b'You WIN!')
else:
    msg = 'You LOSE! The correct word is ' + word
    s.sendall(msg.encode())

s.close()
server.close()
```

```
# socket client
import socket
s = socket.socket()
s.connect(('127.0.0.1', 12345))

print("Guess a word of 5 distinct letters!")
print()
for i in range(5):
    letter = input("Guess a letter in the word:")
    s.sendall(letter.encode())
    reply = s.recv(1024).decode()
    print(reply)

print()
```

```
guess = input("Guess the word:")
s.sendall(guess.encode())
print()
print(s.recv(1024).decode())
s.close()
```

Q2

```
# Task 2.1

# Allow direct copy of contents from json file
import pymongo, json
client = pymongo.MongoClient("127.0.0.1", 27017)
with open("STAFF.json") as file:
    data = json.load(file)
client['Vaccine']['Staff'].insert_many(data)
```

```
# Task 2.2
import pymongo
client = pymongo.MongoClient("127.0.0.1", 27017)
db = client.get_database("Vaccine")
coll = db.get_collection("Staff")

# Display the Name, Age and Reason for staff
# who do not accept to take the vaccine
result = coll.find({"Acceptance":"No"})
print("Staff Who do not accept to take the vaccine:")
print("Name\t\tAge\tReason")
for record in result:
    print("{}\t{}\t{}".format(record["Name"],  record["Age"],
record["Reason"]))
print()


# Display the number of staff with at least one dose of vaccine
taken
result = coll.count({"VacDates":{"$exists":True}})
print("Number of staff with at least one dose of vaccine
taken:", result)
print()


# Add Charlie Lee’s second dose on 5 Apr 2021 and display his
updated record
date = coll.find({"Name":"Charlie Lee"})[0]["VacDates"]
coll.update_one({"Name":"Charlie Lee"}, {"$set": {"VacDates":
[date, "5 Apr 2021"]}})
print(coll.find({"Name":"Charlie Lee"})[0])
print()
```

```
# Display the name(s) of staff with at least 50 years old and
completes both doses
print("Staff with at least 50 years old and completes both
doses:")
result=coll.find({"$and":[{"Age":{"$gte":50}},
{"VacDates":{"$exists":True}}]})
for record in result:
    dates = record["VacDates"]
    if len(dates) == 2:
        print(record["Name"])

client.close()
```

**Output**

```
Staff Who do not accept to take the vaccine:
Name            Age     Reason
Sam Chong       34      Allergy
Susan Lim       32      Pregnancy
Tony Sim        54      Complicated health condition

Number of staff with at least one dose of vaccine taken: 5

{'_id': ObjectId('60876ff0772c4a33ec1d88e7'), 'Name': 'Charlie Lee', 'Age': 41, 'Gender': 'M', 'Accept
ance': 'Yes', 'VacDates': ['19 Mar 2021', '5 Apr 2021']}

Staff with at least 50 years old and completes both doses:
Alan Lim
Helen Wong
```

Q3

```
#Task 3.1

class TreeNode:
    def __init__(self, word, left_ptr=None, right_ptr=None):
        self._word = word
        self._left_ptr = left_ptr
        self._right_ptr = right_ptr

    def getWord(self):
        return self._word

    def setWord(self, word):
        self._word = word

    def getLeftPtr(self):
        return self._left_ptr

    def setLeftPtr(self, left_ptr):
        self._left_ptr = left_ptr

    def getRightPtr(self):
        return self._right_ptr

    def setRightPtr(self, right_ptr):
        self._right_ptr = right_ptr

    def __str__(self):
        return '{}'.format(
            str(self._word))
```

```
#Task 3.2
class BinarySearchTree:
    def __init__(self):
        self._root = None

    def add(self, word):
        if self._root is None:
            self._root = TreeNode(word)
        else:
            self._add(self._root, word)

    def _add(self, root, word):
        if(word < root.getWord()):
            if(root.getLeftPtr() != None):
                self._add(root.getLeftPtr(), word)
            else:
                root.setLeftPtr(TreeNode(word))
        elif(word > root.getWord()):
            if(root.getRightPtr() != None):
                self._add(root.getRightPtr(), word)
            else:
                root.setRightPtr(TreeNode(word))


    def _inOrder(self, root):
        nodeList=[]
        if root:  # First recur on left child
            nodeList.extend(self._inOrder(root.getLeftPtr()))
            # then print the data of node
            nodeList.append(root)
            print(root)
            # now recur on right child
            nodeList.extend(self._inOrder(root.getRightPtr()))
        return nodeList



    def inOrder(self):
        nodeList=[]
        if self._root:  # First recur on left child
            nodeList.extend(self._inOrder(self._root.getLeftPtr()))
            # then print the data of node
            nodeList.append(self._root)
            print(self._root)
            # now recur on right child
            nodeList.extend(self._inOrder(self._root.getRightPtr()))
        return nodeList


    def _preOrder (self, root):
        if root:
            print(root)
            if root.getLeftPtr():
                self._preOrder(root.getLeftPtr())
            if root.getRightPtr():
                self._preOrder(root.getRightPtr())


    def preOrder(self):
        print(self._root)
        if self._root.getLeftPtr():
            self._preOrder(self._root.getLeftPtr())
        if self._root.getRightPtr():
            self._preOrder(self._root.getRightPtr())
```

```
#Task 3.3
filename="TEXT.txt"
bst = BinarySearchTree()
with open(filename, 'r') as f:
    lines = f.readlines()
for line in lines:
    words=line.split()
    for w in words:
       bst.add(w)

#test
print("call in-order:")
lst = bst.inOrder()

OUTPUT:
advance
and
as
civilization
consume
create
more
people
technology
than
they
```

```
#Task 3.4
def balancedBST(lst, l, r, bst):
    mid = l + ((r - l) // 2)
    bst.add(lst[mid].getWord())
    if l<=mid-1:
        balancedBST(lst, l, mid-1, bst)
    if r>=mid+1:
        balancedBST(lst, mid+1, r, bst)
    return bst

#Test
balanced = BinarySearchTree()
balanced = balancedBST(lst, 0, len(lst)-1, balanced)
print()
balanced.preOrder()


OUTPUT:
create
as
advance
and
civilization
consume
technology
more
people
than
they
```