1. What is Abstraction? Why do we need it? What are the two aspects of Abstraction?

- Deciding which details to highlight and which to ignore.

- It allows us to simplify things.

- a) Distill what is essential

- b) Hierarchies that allow different levels of Detail

2. When choosing a password for online accounts, there are typically certain requirements for the strength of the password. Develop a Python program for testing if a string satisfies some appropriate criteria for a strong password. Requirements: Length, Capitals, Numbers, Special Characters.

standard code

print("Password must contain at least 7 characters, a Capital Letter, Number, and Special Character.")

user\_password = input("Enter password.")

if not any(char.isdigit() for char in user\_password):

print("nodigit")

elif not any(char.isupper() for char in user\_password):

print("noupper")

elif not any(char in "!@#$" for char in user\_password):

print("nospec")

else:

break

print("Password accepted.")

print(user\_password)

3. Consider a system for storing anonymous grades of each lab class. Define a Data Structure, which can identify individuals in each lab group by an ID number (1-40 inclusive). To identify the person in the entire class you would also need the group name, eg. ‘FE2’. Each corresponding person should have a number between 1-100 (inclusive) to define grade.

standardcode

def read\_data(data):

user\_data = input(“Enter the student’s “ + data + “. (Or a blank entry to stop.)”)

return user\_data

grades = {}

while True:

student\_group = read\_data(“group name”)

if student\_group == ““:

break

student\_ID = read\_data(“ID”)

grades[student\_group, student\_ID] = read\_data(“grade”)

print(“Student “ + student\_ID + “ data has been entered.”)

print(grades)

4. Given two lists of grades (list of integers) from two classes, write a Python program that will check which class has the highest average score and the highest maximum score.

standardcode

def maxGrade(dataBase, group):

scores = listGrades(dataBase,group)

return max(scores)

5. Write a Python program, in the fewest number of lines possible, which creates a list of all the square numbers: x2 (where 1<=x<=100) that are divisible by 3.

standard code

print[i\*\*2 for i in range(101) if i\*\*2%3 == 0]

6. Develop a Python function inputRecord(dataBase, group, id, score) for TAs to enter one record, where dataBase is the database implemeneted by your data structure. group is a string representing a group name, id is a student’s ID number (positive integers 1-40 inclusive), and score is the grade of the student.

standardcode

def read\_data(data):

user\_data = input(“Enter the student’s “ + data + “. (Or a blank entry to stop.)”)

return user\_data

grades = {}

while True:

student\_group = read\_data(“group name”)

if student\_group == ““:

break

student\_ID = read\_data(“ID”)

grades[student\_group, student\_ID] = read\_data(“grade”)

print(“Student “ + student\_ID + “ data has been entered.”)

print(grades)

7. Develop a Python function query(dataBase, group, id) for TAs to get the score of a student in a lab group, where dataBase is the database implemeneted by your data structure, group is a strong representing a group name, and id is the student’s ID. This function should return the score.

8. Develop a Python function listGrades(dataBase, group) for TAs to get all the student grades in a group, where dataBase is the database implemented by your data structure and group is a string representing a group name. This function should return a list of grades in the group.

9. Develop a Python function maxGrade(dataBase, group) for TAs to get the highest grade in a group, where dataBase is the database implemeneted by your data structure and group is a string representing a group name. This function should return the highest grade.