Questions:

1. **Write a Python program that reads the file created as demonstrated into a dictionary taking ‘name’ as the key and a list consisting of ‘dept’ and ‘cgpa’ as the value for each line. Make changes in some ‘cgpa’ and then write back the whole file.**

2. **Implement in generic ways (as multi-modular and interactive systems) the Greedy Best-First and A* search algorithms in Prolog and in Python.**

Solution to the question no 1

The *Input values* for this given problem are stored in the **“input.txt”** file. This file contains the following values as inputs,

```
Adit MAR 3.43
Anik EEE 3.50
Arnob CSE 2.99
```

The file named **“main.py”** contains the code for the problem. The contents of this file are:

```python
import random

def semesterFinal(name): #takes the name of a student as a parameter
    myList = []
    myList = studentInfo.get(name)
    dept = myList[0]
    cgpa = float(myList[1])
    coeff1 =  random.choice((-1, 1))
    coeff2 = random.uniform(0, 1)
    cgpa = cgpa + (coeff1 * coeff2) #randomly changes the cgpa of a student
    cgpa = round(cgpa,2)
    cgpaStr = str(cgpa)
    studentInfo[name] = (dept,cgpa)


#Main

studentInfo = {}
names = []

file = open("input.txt","r")

for line in file:
    (key,val1,val2) = line.split()
    names.append(key)
    studentInfo[key] = (val1,val2)


```

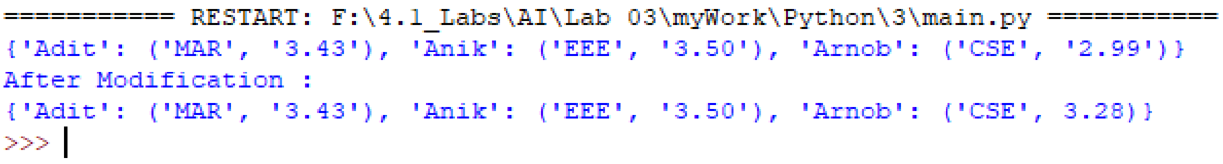
```python
semesterFinal("Arnob")
file.close()


file = open("output.txt","w")

for key in studentInfo:
    myList2 = []
    myList2 = studentInfo.get(key)
    file.write(key)
    file.write(" ")
    file.write(myList2[0])
    file.write(" ")
    file.write(str(myList2[1]))
    file.write("\n")
    file.close()

```

After the execution of **"main.py"** the modified contents are saved onto **"output.txt"** file and momentarily it contains,

```
Adit MAR 3.43
Anik EEE 3.50
Arnob CSE 3.28
```

The **cgpa** of **"Arnob"** has changed after the execution of the code.

A sample output using just the Print operation is demonstrated below:



```
=========== RESTART: F:\4.1_Labs\AI\Lab 03\myWork\Python\3\main.py ===========
{'Adit': ('MAR', '3.43'), 'Anik': ('EEE', '3.50'), 'Arnob': ('CSE', '2.99')}
After Modification :
{'Adit': ('MAR', '3.43'), 'Anik': ('EEE', '3.50'), 'Arnob': ('CSE', 3.28)}
>>> 
```

Solution to the question no 2

For this particular problem, a graph is taken as input. In this case, the graph is represented as a **dictionary** which stores the neighbors as well as their distances from each node. The file **"input.py"** contains this information as well as other initializing factors.

The contents of this file are:

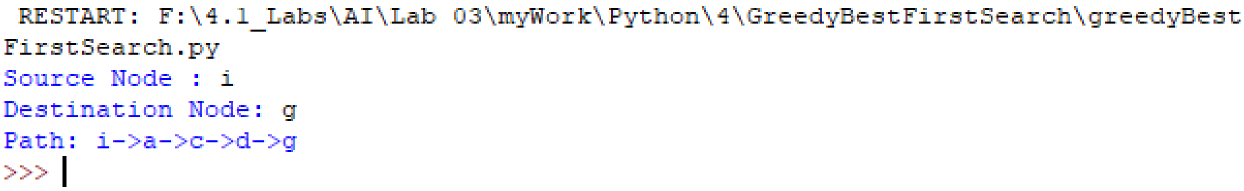
```python
graph = {
    'a': [('c', 22), ('d', 32)],
    'b': [('d', 28), ('e', 36), ('f', 27)],
    'c': [('d', 31), ('g', 47)],
    'd': [('g', 30)],
    'e': [('g', 26)],
    'f': [],
    'g': [],
    'i': [('a', 35), ('b', 45)],
}
```

```
visited = {
    'i': False,
    'a': False,
    'b': False,
    'c': False,
    'd': False,
    'e': False,
    'f': False,
    'g': False
}

heuristic = {
    'i': 80,
    'a': 55,
    'b': 42,
    'c': 34,
    'd': 25,
    'e': 20,
    'f': 17,
    'g': 0
}
```

The code for *Greedy Best First Search* in python that works on this Input, is,

```
import queue as Q
import input as init

pq = Q.PriorityQueue()

def greedyBestFirstSearch():
    source = input("Source Node : ")
    destination = input("Destination Node: ")
    init.visited[source] = True
    pq.put((source, init.heuristic[source], source)) # pq -> (i,80,i)

    while not(pq.empty()):
        u = pq.get() # u -> (i,80,i)
        init.visited[u] = True #init.visited[i] -> true
        if (u[0] == destination): # i != g
            print('Path: ' + u[2]) #wont be executed
            return
        for v in init.graph[u[0]]: # for v in init.graph[u]
            # print("graph of u is : ")
            # print(init.graph[u[0]])
            # print("\n")
            if not init.visited[v[0]]:
                pq.put((v[0], init.heuristic[v[0]], u[2] + '->' + v[0]))

    print("Path not found!")


#Main
greedyBestFirstSearch()
```

A sample input and output would be,

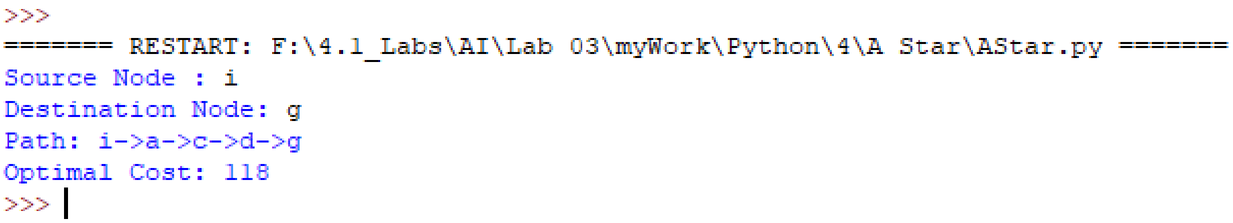


```
 RESTART: F:\4.1_Labs\AI\Lab 03\myWork\Python\4\GreedyBestFirstSearch\greedyBest
FirstSearch.py
Source Node : i
Destination Node: g
Path: i->a->c->d->g
>>> 
```

The code for *A* Search* in python that works on the same Input, is,

```python
import queue as Q
import input as init

pq = Q.PriorityQueue()

def AStar():
    source = input("Source Node : ")
    destination = input("Destination Node: ")
    pq.put((source, init.heuristic[source], source, 0, 0 + init.heuristic[source]))

    while not(pq.empty()):
        u = pq.get()
        if (u[0] == destination):
             print('Path: ' + u[2])
             print('Optimal Cost: ' + str((u[3] + u[4])))
             return
        for v in init.graph[u[0]]:
            pq.put((v[0], init.heuristic[v[0]], u[2] + '->' + v[0], u[3] + v[1], init.heuristic[v[0]]))

    print("Path not found!")

#Main
AStar()
```

A sample output would be,



```
>>> 
======= RESTART: F:\4.1_Labs\AI\Lab 03\myWork\Python\4\A Star\AStar.py =======
Source Node : i
Destination Node: g
Path: i->a->c->d->g
Optimal Cost: 118
>>> 
```