At first we need to extract data from file and for that we need so we will use the simple read function to extract data from the file

```
file = open(r"C:\\Users\\hassa\\OneDrive\\Desktop\\AI\\file.txt")
```

Then we will one by one store the values of the M. N and T.
Then for one empty line we will do a file.readline()

Then we will read all the remaining lines in a list called “line”

```
m=int(file.read(1))
n=int(file.read(2))
t=int(file.read(3))
file.readline()
lines=file.readlines()
```

Now, we need to access the data by indices. Since, we have “m” number of states, so we will read from line 0 to line number “m”
Since we are reading strings we will have to remove the “endlines” and “tabs”.

```
states=lines[0:m]
for x in range(len(states)):
    states[x]=states[x].replace("\n","")
    states[x]=states[x].replace("\t","")
```

Now, we will read data for actions that we are given and for that we will read data from where we left off and towards the length of “n” we are doing +1 because after “m” number of states we need to skip on line.

```
actions=lines[m+1:m+1+n]
for x in range(len(actions)):
    actions[x]=actions[x].replace('\n','')
    actions[x]=actions[x].replace('\t','')
```

Now, we will read data for the transition matrix. And we will add the indices of the already read data and add 2 because we skipped one line after “m” and another after “n”

```
matrix=lines[m+2+n:m+2+n+m]
for x in range(len(matrix)):
    matrix[x]=matrix[x].replace('\n','')
    matrix[x]=matrix[x].replace('\t','')
```

Now, we need to read test cases and we will add all the number of lines read before and towards the length of the test cases provided to us in the first line of the input.

```
testcases=lines[m+n+m+3:m+n+m+t+3]
for x in range(len(testcases)):
    testcases[x]=testcases[x].replace('\n','')
```

There is an issue all the cases that we were given below were separated by “endlines” but here we also have “Tabs” here.
Right now, each index contain one string but it has two test cases in that string separated by tab like {‘Stringa \t stringB’}
We will iterate through the list and split each index with respect to \t and then we will store both of test cases as separate indices in a different list called tcase. Tcase will append 2 on every iteration of every x

```
for x in range(len(testcases)):
    string=testcases[x].split('\t')
    for i in range(len(string)):
        tcase.append(string[i])
```

Now, we need to find the a way to convert matrix in to 3d array with numerics instead of string format. For that, we iterated through the list

where each index contain n numbers so we will first split each of them by space and store then in list2.
{‘1 2 3’ , ‘5 4 6’} has become {‘1’, ‘2’, ‘3’ …} and now we will go through the each of this as individual and add them to another list after converting them to ints

```
list2=[]
for x in range(len(matrix)):
    for y in range(len(matrix[x])):
        list2=matrix[x].split(' ')
    for z in range(len(list2)):
        stack1.append(int(list2[z]))
```

Now, we have ourselves a numeric array and we need to convert it into m x n matrix, For that we need include a numpy library and use a function called “reshape”

```
matrix=np.reshape(stack1,(m,n))
```

Now, we need to implement the logic where we do the searching

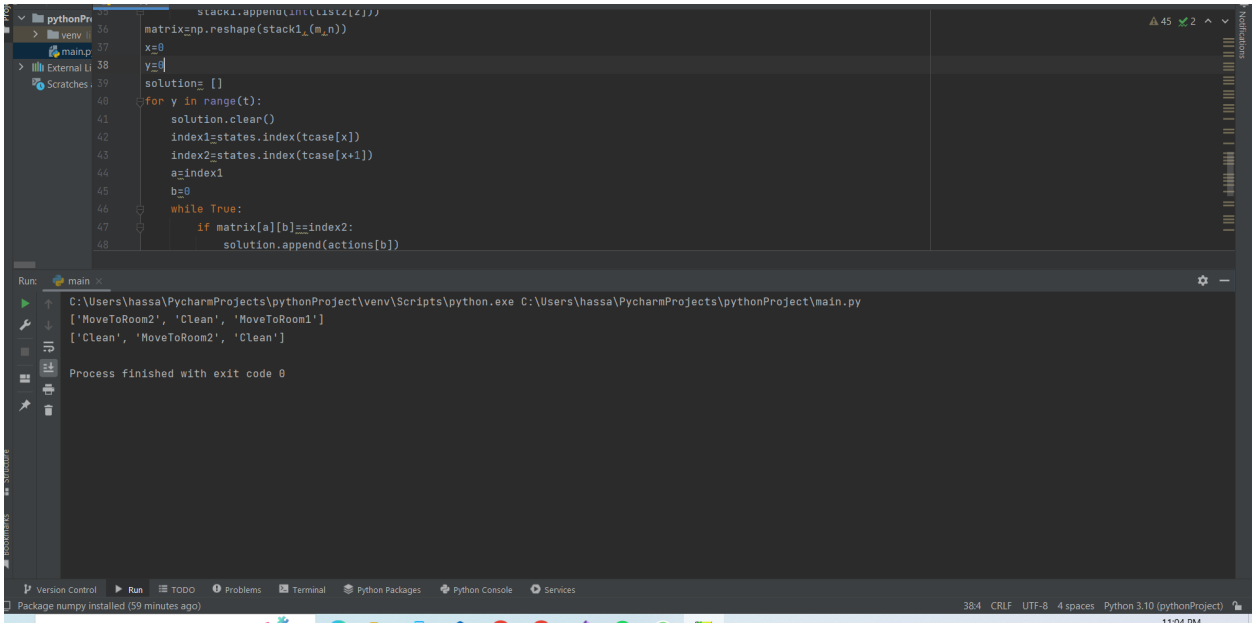
First we need a loop that will run till all the test cases are met and then We need a loop for each test case.

```
for y in range(t):
    solution.clear()
    index1=states.index(tcase[x])
    index2=states.index(tcase[x+1])
    a=index1
    b=0
    while True:
        if matrix[a][b]==index2:
            solution.append(actions[b])
            break
        elif a==matrix[a][b]:
            b=b+1
        else:
            solution.append(actions[b])
            a=matrix[a][b]
            b=0
    x=x+2
    print(solution)
```

First I will store two test cases, first that is given and the other one which is required.
Then we will run an infinite loop which will not stop untill the required test case is met.
First if condition will check if we are already at the solution. And it will break the inner loop and will go the next test case.
Second if condition will check if after performing an action we stay in the same state so we will move right in matrix and basically do nothing
Last condition will run if actually need to perform a successful action for that we will move to the test case whose index we encounter in the matrix,
We are making b =0 because if go to a new row of matrix we will have to check from starting,

After each iteration of the loop we are doing x=x+2 because every 2 consecutive indices contain given and required test cases.
We are storing the steps we are doing in a list and printing it after every iteration and also clearing it before start of every outer loop iteration.

Now, we will evaluate some outputs.
The given input file.



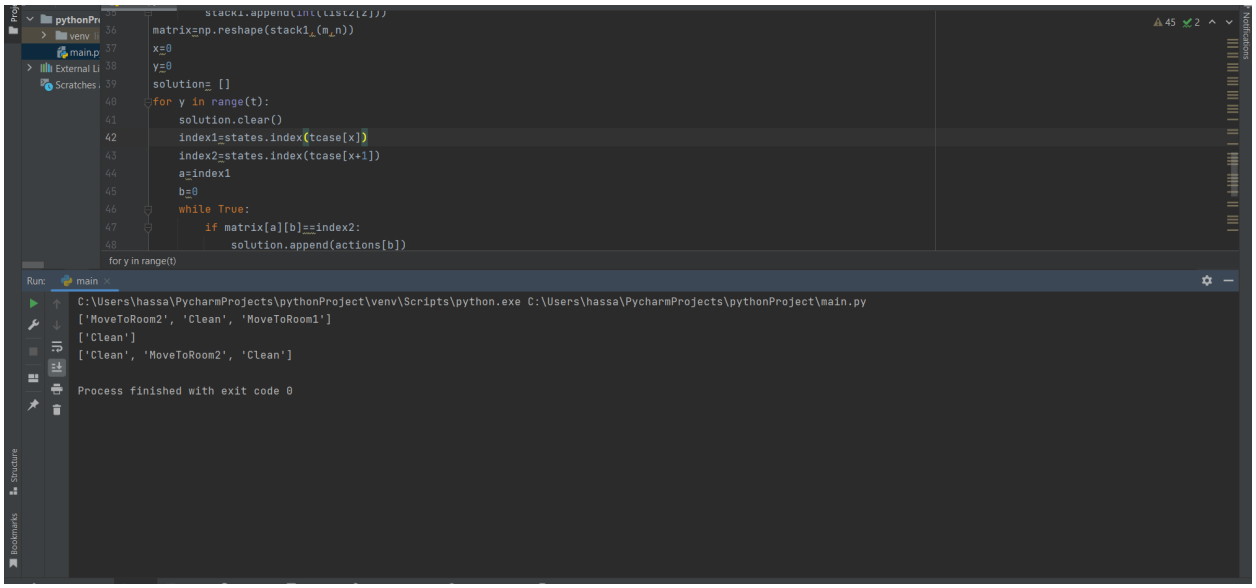
Now, we will increase a test case and increase from 2 to 3

file - Notepad

File Edit Format View Help

```
8 3 3

(AgentInRoom1, Room1Dusty, Room2Dusty)
(AgentInRoom1, Room1Dusty, Room2Clean)
(AgentInRoom1, Room1Clean, Room2Dusty)
(AgentInRoom1, Room1Clean, Room2Clean)
(AgentInRoom2, Room1Dusty, Room2Dusty)
(AgentInRoom2, Room1Dusty, Room2Clean)
(AgentInRoom2, Room1Clean, Room2Dusty)
(AgentInRoom2, Room1Clean, Room2Clean)

Clean
MoveToRoom1
MoveToRoom2

2 0 4
3 1 5
2 2 6
3 3 7
5 0 4
5 1 5
7 2 6
7 3 7

(AgentInRoom1, Room1Clean, Room2Dusty)  (AgentInRoom1, Room1Clean, Room2Clean)
(AgentInRoom2, Room1Clean, Room2Dusty)  (AgentInRoom2, Room1Clean, Room2Clean)
(AgentInRoom1, Room1Dusty, Room2Dusty)  (AgentInRoom2, Room1Clean, Room2Clean)
```

Now, we will run the same program.



Now, let us try with 5

```
File  Edit  Format  View  Help
8 3 5

(AgentInRoom1, Room1Dusty, Room2Dusty)
(AgentInRoom1, Room1Dusty, Room2Clean)
(AgentInRoom1, Room1Clean, Room2Dusty)
(AgentInRoom1, Room1Clean, Room2Clean)
(AgentInRoom2, Room1Dusty, Room2Dusty)
(AgentInRoom2, Room1Dusty, Room2Clean)
(AgentInRoom2, Room1Clean, Room2Dusty)
(AgentInRoom2, Room1Clean, Room2Clean)

Clean
MoveToRoom1
MoveToRoom2

2 0 4
3 1 5
2 2 6
3 3 7
5 0 4
5 1 5
7 2 6
7 3 7

(AgentInRoom1, Room1Clean, Room2Dusty)  (AgentInRoom1, Room1Clean, Room2Clean)
(AgentInRoom2, Room1Clean, Room2Dusty)  (AgentInRoom2, Room1Clean, Room2Clean)
(AgentInRoom1, Room1Dusty, Room2Dusty)  (AgentInRoom2, Room1Clean, Room2Clean)
(AgentInRoom1, Room1Dusty, Room2Dusty)  (AgentInRoom2, Room1Clean, Room2Clean)
(AgentInRoom1, Room1Clean, Room2Dusty)  (AgentInRoom1, Room1Clean, Room2Dusty)
```

Here we have an issue even though the given and final state are equal we are still performing an action. To do that we will use another if condition It will check if the given and required state are equal or index1==index2.

```
    a=index1
    b=0
    while True:
        if index1==index2:
            break
        elif matrix[a][b]==index2:
            solution.append(actions[b])
            break
        elif a==matrix[a][b]:
            b=b+1
        else:
            solution.append(actions[b])
            a=matrix[a][b]
            b=0
    x=x+2
    print(solution)
file.close()
```

When we run program again,

```
for y in range(t)  ›  while True  ›  if index1==index2
Run:  main
C:\Users\hassa\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\hassa\PycharmProjects\pythonProject\main.py
['MoveToRoom2', 'Clean', 'MoveToRoom1']
['Clean']
['Clean', 'MoveToRoom2', 'Clean']
['Clean', 'MoveToRoom2', 'Clean']
[]

Process finished with exit code 0
```

Now, we have an empty set for the index1==index2

Now we need to add arrows in the output. We will first convert it into string and then we will remove the commas with arrows and remove the braces.
There is one case where we might get infinite loop when two rows of matrix keep sending to each other. We will just make an explored set list which will keep record of what has been already processed.

```
        b=b+1
    elif a in explored:
        b=b+1
```