Importing the required packages

In [3]:
```
import numpy as np
import itertools
import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
```

In [4]:
```
%%file coursesheet.csv
Rno,Department,Year,Course:1,Course:2,Course:3,Course:4,Course:5,Course:6
1,CS,1,GT,CA,DS,BEE,DE,NaN
2,CS,1,GT,CA,DS,BEE,DE,NaN
3,CS,1,GT,CA,DS,BEE,DE,NaN
4,CS,1,GT,CA,DS,BEE,DE,NaN
5,CS,1,GT,CA,DS,BEE,DE,NaN
6,CS,2,CD,FDB,CAA,CAE,NaN,NaN
7,CS,2,CD,CA,CAA,CAE,DE,NaN
8,CS,2,DE,CA,CAA,CAE,CD,FDB
9,CS,2,CD,CA,CAA,CAE,ITE,NaN
10,CS,2,CD,SME,CAA,CAE,FDB,NaN

1,IT,1,GT,CA,DS,BEE,NaN,NaN
2,IT,1,GT,CA,DS,BEE,NaN,NaN
3,IT,1,GT,CA,DS,BEE,NaN,NaN
4,IT,1,GT,CA,DS,BEE,NaN,NaN
5,IT,1,GT,CA,DS,BEE,NaN,NaN
6,IT,2,SME,ITG,ITF,ITU,ITH,NaN
7,IT,2,ITG,DS,ITF,ITH,GT,ITU
8,IT,2,ITF,ITG,ITU,ITH,NaN,NaN
9,IT,2,ITG,ITF,ITU,SME,ITH,NaN
10,IT,2,ITF,ITU,ITH,NaN,ITG,GT

1,EEE,1,GT,CA,DS,BEE,NaN,NaN
2,EEE,1,GT,CA,DS,BEE,NaN,NaN
3,EEE,1,GT,CA,DS,BEE,NaN,NaN
4,EEE,1,GT,CA,DS,BEE,NaN,NaN
5,EEE,1,GT,CA,DS,BEE,NaN,NaN
6,EEE,2,EER,EET,DS,EEJ,ITE,NaN
7,EEE,2,EER,CA,DS,EEJ,ITE,EET
8,EEE,2,EER,CA,EEJ,SME,ITE,EET
9,EEE,2,EER,SME,EEJ,ITE,EET,NaN
10,EEE,2,EER,CA,EEJ,ITE,EET,NaN
```

```
Overwriting coursesheet.csv
```

Creating a dataframe to display the course details

In [5]:
```
df=pd.read_csv("coursesheet.csv")
df
```

Out[5]:

	Rno	Department	Year	Course:1	Course:2	Course:3	Course:4	Course:5	Course:6
0	1	CS	1	GT	CA	DS	BEE	DE	NaN
1	2	CS	1	GT	CA	DS	BEE	DE	NaN
2	3	CS	1	GT	CA	DS	BEE	DE	NaN
3	4	CS	1	GT	CA	DS	BEE	DE	NaN
4	5	CS	1	GT	CA	DS	BEE	DE	NaN
5	6	CS	2	CD	FDB	CAA	CAE	NaN	NaN
6	7	CS	2	CD	CA	CAA	CAE	DE	NaN
7	8	CS	2	DE	CA	CAA	CAE	CD	FDB
8	9	CS	2	CD	CA	CAA	CAE	ITE	NaN
9	10	CS	2	CD	SME	CAA	CAE	FDB	NaN
10	1	IT	1	GT	CA	DS	BEE	NaN	NaN
11	2	IT	1	GT	CA	DS	BEE	NaN	NaN
12	3	IT	1	GT	CA	DS	BEE	NaN	NaN
13	4	IT	1	GT	CA	DS	BEE	NaN	NaN
14	5	IT	1	GT	CA	DS	BEE	NaN	NaN
15	6	IT	2	SME	ITG	ITF	ITU	ITH	NaN
16	7	IT	2	ITG	DS	ITF	ITH	GT	ITU
17	8	IT	2	ITF	ITG	ITU	ITH	NaN	NaN
18	9	IT	2	ITG	ITF	ITU	SME	ITH	NaN
19	10	IT	2	ITF	ITU	ITH	NaN	ITG	GT
20	1	EEE	1	GT	CA	DS	BEE	NaN	NaN
21	2	EEE	1	GT	CA	DS	BEE	NaN	NaN
22	3	EEE	1	GT	CA	DS	BEE	NaN	NaN
23	4	EEE	1	GT	CA	DS	BEE	NaN	NaN
24	5	EEE	1	GT	CA	DS	BEE	NaN	NaN
25	6	EEE	2	EER	EET	DS	EEJ	ITE	NaN
26	7	EEE	2	EER	CA	DS	EEJ	ITE	EET
27	8	EEE	2	EER	CA	EEJ	SME	ITE	EET
28	9	EEE	2	EER	SME	EEJ	ITE	EET	NaN
29	10	EEE	2	EER	CA	EEJ	ITE	EET	NaN

Reading the input csv file and listing out the branches and years

In [6]:
```
data = pd.read_csv("coursesheet.csv", delimiter = ',')
df = pd.DataFrame(data)
```

Displaying the Departments in the data

In [8]:
```
Dept = df['Department'].unique()
DeptList= len(Dept)
print(f"Department:{Dept}")
```

```
Department:['CS' 'IT' 'EEE']
```

Displaying the Years in the data

In [10]:
```
year = df['Year'].unique()
yearlist= len(year)
print(f"\nYear:{year}")
```

```
Year:[1 2]
```

We created a dictionary called courses and intially it is empty variable, var is created and its value is given as 0

In [12]:
```
courses={}
var=0
```

We are using the keys and value pairs in the dictionary to update the courses.

In [26]:
```
for c1,c2,c3,c4,c5,c6,c in zip(df['Course:1'],df['Course:2'],df['Course:3'],df['Course:
    if c1 not in courses and c1 == c1:
        courses.update({c1:var})
        var=var+1
    if c2 not in courses and c2 == c2:
        courses.update({c2:var})
        var=var+1
    if c3 not in courses and c3 == c3:
        courses.update({c3:var})
        var=var+1
    if c4 not in courses and c4 == c4:
        courses.update({c4:var})
        var=var+1
    if c5 not in courses and c5 == c5:
        courses.update({c5:var})
        var=var+1
    if c6 not in courses and c6 == c6:
        courses.update({c6:var})
```

```
        var=var+1

print("\n Number of courses :",len(courses))

print("\n List of courses:",courses)
```

```
 Number of courses : 18

 List of courses: {'GT': 0, 'CA': 1, 'DS': 2, 'BEE': 3, 'DE': 4, 'CD': 5, 'FDB': 6, 'CA
A': 7, 'CAE': 8, 'ITE': 9, 'SME': 10, 'ITG': 11, 'ITF': 12, 'ITU': 13, 'ITH': 14, 'EER':
15, 'EET': 16, 'EEJ': 17}
```

Creating a matrix for the courses

In [56]:

```
matrix = [[0 for i in range(len(courses))] for j in range(yearlist*DeptList)]
i=0
tmp=Dept[0]


for c1,c2,c3,c4,c5,c6,year,course in zip(df['Course:1'],df['Course:2'],df['Course:3'],d
    if tmp==course:
        year=year+i*yearlist
    else:
        i=i+1
        tmp=Dept[i]
        year=year+i*yearlist
    if c1 == c1:
        matrix[year-1][courses[c1]]=1
    if c2 == c2:
        matrix[year-1][courses[c2]]=1
    if c3 == c3:
        matrix[year-1][courses[c3]]=1
    if c4 == c4:
        matrix[year-1][courses[c4]]=1
    if c5 == c5:
        matrix[year-1][courses[c5]]=1
    if c6 == c6:
        matrix[year-1][courses[c6]]=1


SMatrix=pd.DataFrame(matrix, columns=courses.keys())
print("\n\n Year wise list of courses:")
SMatrix
```

```
 Year wise list of courses:
```

Out[56]:

	GT	CA	DS	BEE	DE	CD	FDB	CAA	CAE	ITE	SME	ITG	ITF	ITU	ITH	EER	EET	EEJ
0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0
2	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
4	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	GT	CA	DS	BEE	DE	CD	FDB	CAA	CAE	ITE	SME	ITG	ITF	ITU	ITH	EER	EET	EEJ
5	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1

In [28]:
```
rvs = dict(zip(courses.values(),courses.keys()))
print(rvs)
```

```
{0: 'GT', 1: 'CA', 2: 'DS', 3: 'BEE', 4: 'DE', 5: 'CD', 6: 'FDB', 7: 'CAA', 8: 'CAE', 9:
'ITE', 10: 'SME', 11: 'ITG', 12: 'ITF', 13: 'ITU', 14: 'ITH', 15: 'EER', 16: 'EET', 17:
'EEJ'}
```

In [30]:
```
Max=0
for i in range(0,DeptList*yearlist):
    count=0
    for j in range(len(courses)):
        if matrix[i][j]==1:
            count+=1
    if(count>Max):
        Max=count

print(Max)
```

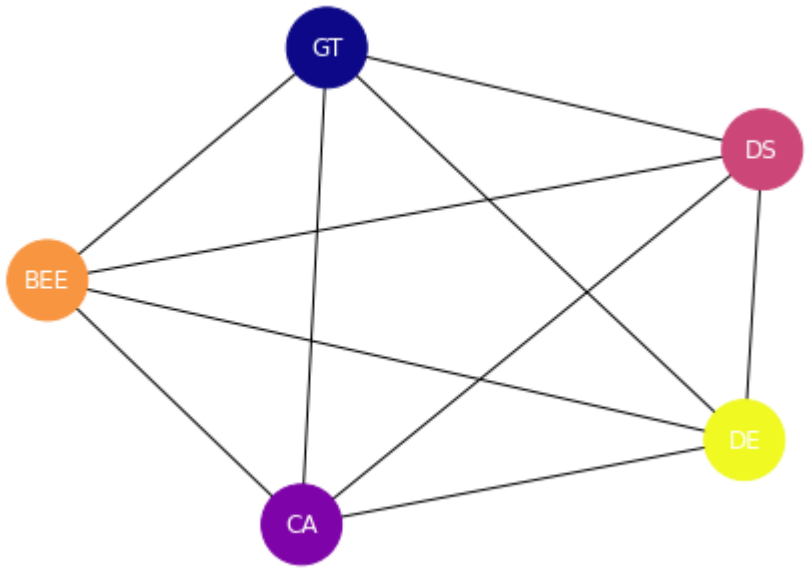
```
8
```

Displaying graph for every year:

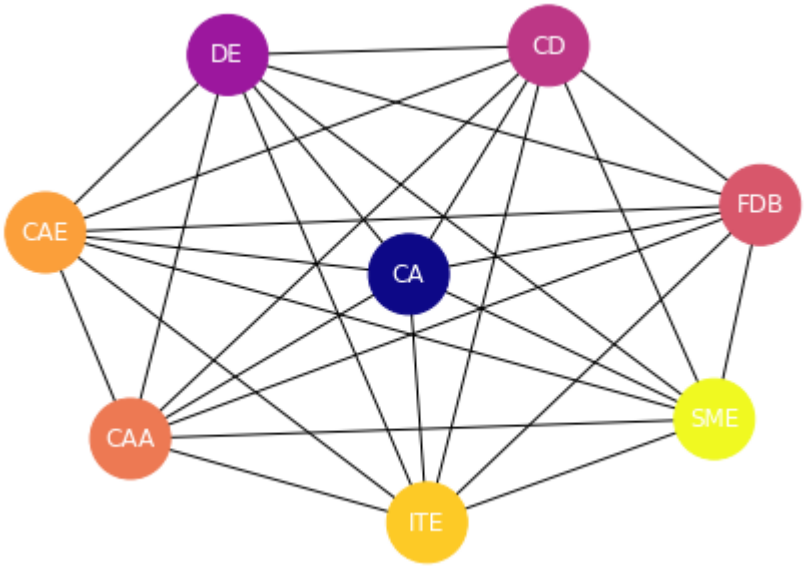
In [45]:
```
Max=2*Max
CoursesColor={}
DataSheet=[]
chromatic=[]
TempChrom=[]
CGraph = nx.Graph()
x1=1

for j1 in range(0,Max):
    TempChrom.append(j1)

for i1 in range(0,DeptList*yearlist):
    Subject=[]
    G = nx.Graph()
    for j1 in range(0,len(courses)):
        if matrix[i1][j1]==1:
            Subject.append(rvs[j1])
    DataSheet.append(Subject)


    chromatic=TempChrom


    for y1 in range(0,i1):
        for z1 in range(0,len(courses)):
            if matrix[y1][z1] == 1 and rvs[z1] in Subject and CoursesColor.get(rvs[z1])
                chromatic.remove(CoursesColor[rvs[z1]])

    for y1 in range(i1+1,yearlist*DeptList):
```

```
        for z1 in range(0,len(courses)):
            if matrix[y1][z1] == 1 and rvs[z1] in CoursesColor.keys() and CoursesColor.
                chromatic.remove(CoursesColor[rvs[z1]])


    index=0
    for Subjectject in range(0,len(Subject)):
        if Subject[Subjectject] not in CoursesColor.keys():
            CoursesColor.update({Subject[Subjectject]:chromatic[index]})
            index=index+1

    print("Graph for year",i1+1,":")
    print(Subject)
    x1+=1
    G.add_nodes_from(Subject)

    G.add_edges_from(itertools.combinations(Subject, 2))
    val = [CoursesColor.get(node,0.25) for node in Subject]
    CGraph.add_nodes_from(Subject)
    CGraph.add_edges_from(itertools.combinations(Subject, 2), weight =8)

    nx.draw(G, node_size=1600,cmap=plt.get_cmap('plasma'), node_color=val, with_labels=

    plt.show()
```

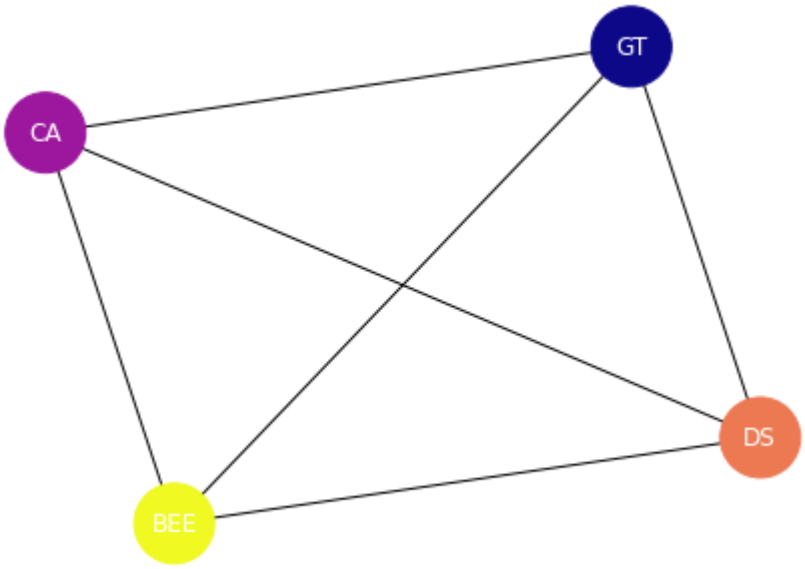
```
Graph for year 1 :
['GT', 'CA', 'DS', 'BEE', 'DE']
```



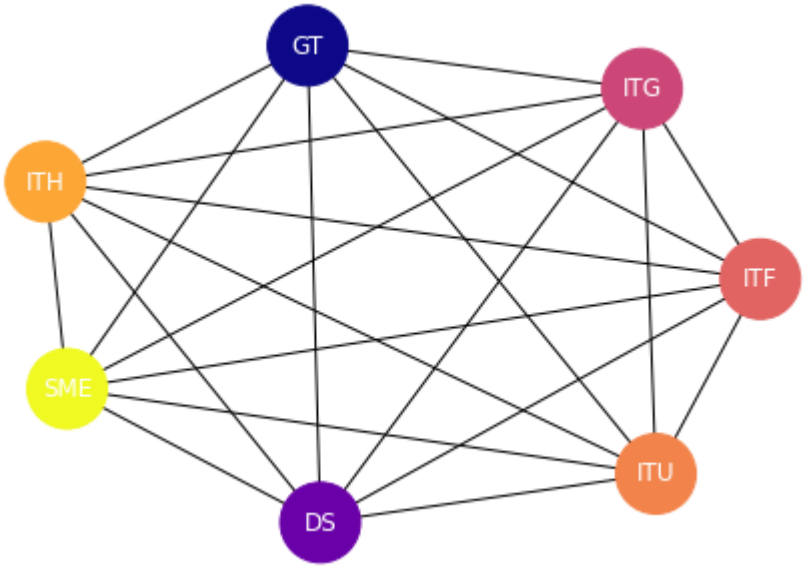

```
Graph for year 2 :
['CA', 'DE', 'CD', 'FDB', 'CAA', 'CAE', 'ITE', 'SME']
```



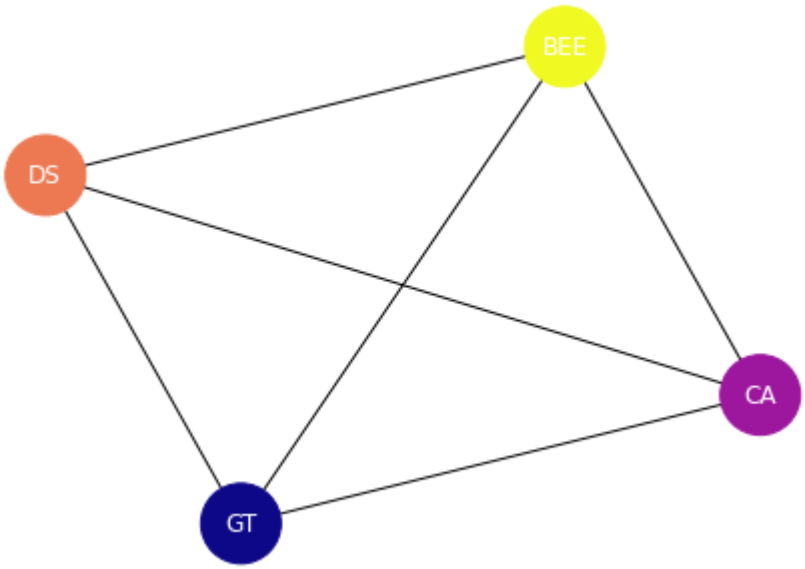
Graph for year 3 :
['GT', 'CA', 'DS', 'BEE']



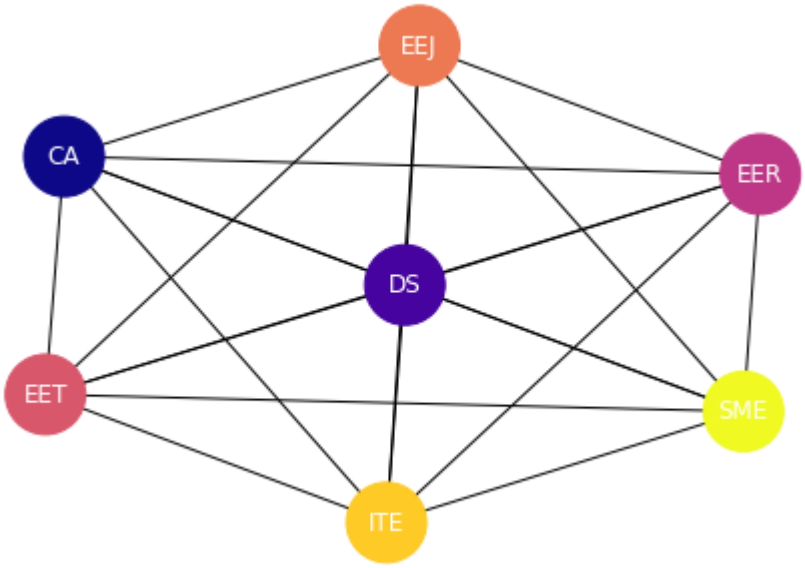
Graph for year 4 :
['GT', 'DS', 'SME', 'ITG', 'ITF', 'ITU', 'ITH']



Graph for year 5 :
['GT', 'CA', 'DS', 'BEE']



Graph for year 6 :
['CA', 'DS', 'ITE', 'SME', 'EER', 'EET', 'EEJ']

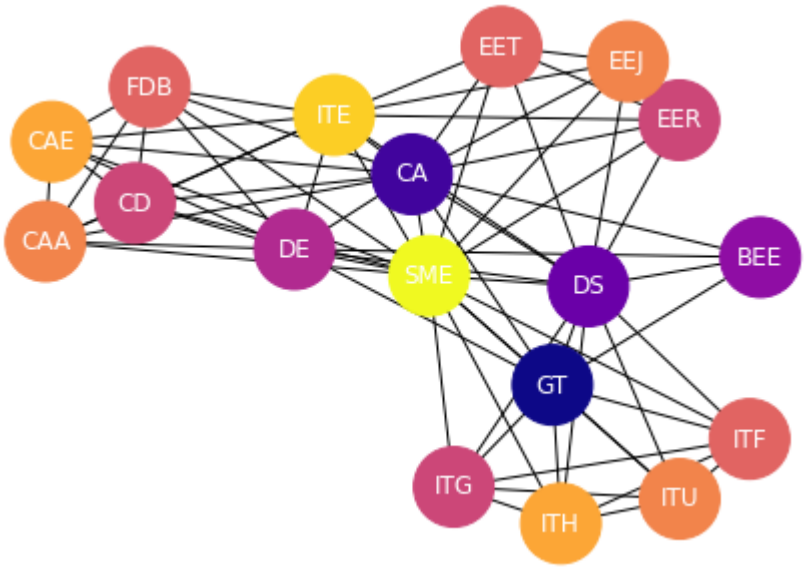


Displaying Graph for all the courses:

In [44]:

```
print("All courses Graph:")
val = [CoursesColor.get(node,0.35) for node in CGraph.nodes()]
nx.draw(CGraph, node_size=1600, cmap=plt.get_cmap('plasma'), node_color=values, with_la
nx.draw
plt.show()
```

```
All courses Graph:
```



Colors assigned to the courses as below:

In [46]:

```
print("Color of the Course:")
for x,y in zip(CoursesColor.keys(),CoursesColor.values()):
    print(x,"-",y)
```

```
Color of the Course:
```

```
GT - 0
CA - 1
DS - 2
BEE - 3
DE - 4
CD - 5
FDB - 6
CAA - 7
CAE - 8
ITE - 9
SME - 10
ITG - 5
ITF - 6
ITU - 7
ITH - 8
EER - 5
EET - 6
EEJ - 7
```

Displaying maximum number of colors used :

In [47]:

```
Max=-1
for i,j in zip(CoursesColor.keys(),CoursesColor.values()):
    if(j>Max):
        Max=j
Max=Max+1
print(Max)
```

```
11
```

Displaying the final exam scheduler developed:

In [59]:

```
temp=Max+2

data=[['']*temp for i in range(DeptList*yearlist)]
column=['Department','Year']


for i in range(0,DeptList*yearlist):
    for j in range(0,len(courses)):
        if matrix[i][j] is 1:
            data[i][2+CoursesColor[rvs[j]]]=str(rvs[j])


for i in range(1,Max+1):
    day='Day'+str(i)
    column.append(day)

finalschedule = pd.DataFrame(data, columns=column)
j=1
for i in range(0,DeptList*yearlist):
    if i < j*yearlist:
        finalschedule.at[i,'Department']=(df['Department'].unique()[j-1])
    else:
```

```
        j=j+1
        finalschedule.at[i,'Department']=(df['Department'].unique()[j-1])

    finalschedule.at[i,'Year']=(df['Year'].unique()[(i)%yearlist])

print("\nFinal exam schedule generated for the given data is:")
finalschedule
```

```
Final exam schedule generated for the given data is:
<>:9: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:9: SyntaxWarning: "is" with a literal. Did you mean "=="?
C:\Users\manas\AppData\Local\Temp/ipykernel_28604/3874309727.py:9: SyntaxWarning: "is" w
ith a literal. Did you mean "=="?
  if matrix[i][j] is 1:
```

Out[59]:

	Department	Year	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11
0	CS	1	GT	CA	DS	BEE	DE						
1	CS	2		CA			DE	CD	FDB	CAA	CAE	ITE	SME
2	IT	1	GT	CA	DS	BEE							
3	IT	2	GT		DS			ITG	ITF	ITU	ITH		SME
4	EEE	1	GT	CA	DS	BEE							
5	EEE	2		CA	DS			EER	EET	EEJ		ITE	SME

In [ ]: