1: Unix Question:

Store details are stored in following format:

StoreId,Name,Type,StoreRevenue, StoreExpenses

Write the unix command(s) to display all the fields mentioned above along with the Store Profit field (As a last field of the every store record) for all the Stores. The output records of multiple stores are to be displayed in the ascending order of profit.

Output format:
StoreId,Name,Type,StoreRevenue, StoreExpenses and StoreProfit (Caluculated as part of the script/command) are to be displayed with a "|" as the delimiter in the output . For more details on format and order of output refer the sample testcase in the below sections

Formula for caluculation of Store profit :
StoreProfit = StoreRevenue - StoreExpenses

If StoreProfit ***is less then*** *Zero, then the store profit is in negative*

The Store details are provided as command line argument when the file containing your command will run. Use appropriate command line argument($1,$2 etc.) to access the details in your file, where you have written the commands.

For more details, please refer to the sample input and output below.

Input Format words in the box used for tom Testing.
Input data (From Test case input or from Custom input field) will be supplied to the shell script through commandline argument.
You just need to use the appropriate commandline argument in the shell script to read the content/input data and store the same into a file .Finally Process the file towards the given requirement.
To test your code with custom input option the respective checkbox needs to be enabled to enter the input data mentioned in the Qn text.

Instructions to read data from custom input text field:
1.The first input line contains the **StoreId,Name,Type,StoreRevenue, StoreExpenses** for the first store and
fields/columns of the Store separated by comma.
2. The subsequent lines contains the details of the rest of the stores provided one by one as mentioned in Point#1.

Refer the sample test case below for more details:

Sample test case
Test case Input

```
1,RamsDeptStore,Stationary,100,50
2,RajStore,Departmental,88,84
3,HealthyStore,Grocery,95,97
4,MiniStore,Medical,60,55
```

Test case output:

```
3|HealthyStore|Grocery|95|97|-2
2|RajStore|Departmental|88|84|4
4|MiniStore|Medical|60|55|5
1|RamsDeptStore|Stationary|100|50|50
```

Output Explanation
For all the stores , Profit is calculated (Revenue - Expenses) and the records are arranged in the ascending order of profit. Ie Least profit(Could be negative / Loss also) store (ie Healthystore from the sample input test case above) among all will be displayed as first record in output and the next least profit store(RajStore) among the remaining stores(other than the least profit store) can be placed as second record in the output. and So On.., In the same line the last record in the output is top profit store

Solution:

```
awk -F ',' '$3=="Python" && $4=="Oracle" {print $1"&"$2"&"$3"&"$4}' $1  > temp.txt
awk '{ print $0 }' temp.txt|sort -t"|" -rk2
```

Python Qn:

Create a class **Student** with the below attributes:

name of type String //specifies name of the student
sub1 of type float //specifies marks of the subject1
sub2 of type float //specifies marks of the subject2
sub3 of type float //specifies marks of the subject3

Create the **__init__** method which takes all parameters in the above sequence. The method should set the value of attributes to parameter values .

Create a method **calculateResult** in the **Student** class, It accept **Student** object as an input parameter, It checks, if the student has scored >40 in all the individual 3 subjects. If so, it further calculates the average and returns the average. The average result will be calculated according to the below formula.
Note : If any student scores < 40 in any of the subject then zero is returned as average.
Formula : average = (sub1+ sub2 + sub3)/3

Create another class **School** with the below attribute:
name of type String //specifies the name of the school
studentDict of type dictionary //specifies the dictionary of students, where key is a **Student** Object and value refers to the result (fail or pass) of that respective student.

Create the **__init__** method which takes all parameters in the above sequence. The method should set the value of attributes to parameter values inside the method.

Define the two methods (**getStudentResult** and **findStudentWithHighestMarks)** in this **School** class as per the below requirement

getStudentResult : This method internally calls **calculateResult** method of **Student** class for every student in the student dictionary (of School class) to get the average of marks of the respective student. This method checks if the student average > 60, then it update the student dictionary **value** for the respective key(respective student object) as "pass" and returns that dictionary of the Student objects to main, where in Student object is a key and Result (pass or fail) as value.

Note: by default the **value** for each of the student objects , acting as a key in the dictionary of students set as “fail” in main function before calling this function.
Then the main function displays the names of students, who passed .

Create another method inside School class with the name **findStudentWithHighestMarks**. This method accept the list of passed students as input from main, this method also internally calls the **calculateResult** method of Student class for every **Student** to get the average for each student and returns a **Student** object with highest average score to main. Main function displays the name of the highest scored student, interms of average.

Note : If there is no student passed (i.e. the input student list is empty) this method returns **None , then the main** program will print “**No Student Passed**”(Excluding the double quotes)

Instructions to write main and to call the methods of the classes defined above:
a. You would required to write the main function completely, please follow the below instructions for the same.
b. You would required to write the main program which is inlign to the "**sample input description section"** mentioned below and **to read the data in the same sequence.**
c. Create the respective objects(**Student** and **School**) with the given sequence of arguments to fulfill the constructor requirement defined in the respective class.
i.Create a Student object after reading the data related to **Student** and add the **Student** object (to **Dictionary of Students** objects) as a key and "fail" as a value. Ie the respective student object and result("fail") as key : value pair.
This point repeats for the number of **Student** objects you want to create and add to **Dictionary of Students ie** as mentioned in the first line of input in the input data.

ii. Create **School o**bject by passing the School name and **Dictionary of Students** objects (created and as mentioned in above point# c.i) as the arguments to the constructor .
d. Call the methods (**getStudentResult** and **findStudentWithHighestMarks)** mentioned above from the main function in the same order, they appear in the question text.
e. Display the data returned by the functions,in the main program as per the order of calling the functions and as per the format mentioned in the sample output and as per Qn.
f. If **None** is returned by **findStudentWithHighestMarks** function then display "**No Student Passed**"(Excluding the double quotes) in the main function.

You can use/refer the below given sample input and output to verify your solution using '**Custom Input** option in Hackerrank.
Sample Input (below) description:
- The first input specifies the count of students.
- Second set of inputs(2nd line to 5th line) are the student name, marks of sub1, marks of sub2, marks of sub3 respectively. This information will be taken for complete set of students, as per the count mentioned in the first line of inputs.
- Last input is the name of the school of type String.

Sample test case input

```
4

armaan
40
55
60
shivam
55
78
90
jai
60
77
94
rajan
10
80
90
DPS
```

Sample testcase output:

```
shivam
jai
jai
```

From three records, only **shivam** and **jai** got the average more than 60 and their individual each subject marks are more than or equals to 40, Hence these two student names are displayed in the first two lines of output.
Last line of output(**jai**) represent the student, who has passed and scored hishest interms of average of three subject marks.

Solution:

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Enter your code here. Read input from STDIN. Print output to STDOUT
class Student:
    def __init__(self,name,sub1,sub2,sub3):
        self.name=name
        self.sub1=sub1
        self.sub2=sub2
        self.sub3=sub3
    def calculateResult(self):
        averageMarks=0
        if(self.sub1>40 and self.sub2>40 and self.sub3>40):
            totalMarks=self.sub1+self.sub2+self.sub3
            averageMarks=round(totalMarks/3,2)
        return averageMarks

class School:
    def __init__(self,name,studentDict={}):
        self.schoolName=name
        self.studentDict=studentDict

    def getStudentResult(self):
        averageMarks=0
        for student in self.studentDict:
            averageMarks=student.calculateResult()
            if averageMarks>60:
                self.studentDict[student]="pass"
        return studentDict

    def findStudentWithHighestMarks(self,lpassedStudentList):
        highestMarks=0
        averageMarks=0
        topStudent=None
        for stud in lpassedStudentList:
            averageMarks=stud.calculateResult()
            if averageMarks > highestMarks:
                highestMarks=averageMarks
                topStudent=stud
        return topStudent

if __name__=='__main__':
    studentDict={}
    noOfStudents=int(input())
    for i in range(noOfStudents):
        name=input()
        sub1=float(input())
        sub2=float(input())
        sub3=float(input())
        result="fail"
```

```
    student=Student(name,sub1,sub2,sub3)
    studentDict[student]=result
schoolName=input()
school=School(schoolName,studentDict)
StuResultDict=school.getStudentResult()
passedStudentList=[]
for s in StuResultDict:
    if StuResultDict[s]=="pass":
        print(s.name)
        passedStudentList.append(s)
topStudent=school.findStudentWithHighestMarks(passedStudentList)
if topStudent!=None:
    print(topStudent.name)
else:
    print('No Student Passed')
```