Submission instruction:

The student will work on a programming assignment using an online system. The student will receive an invite to an online coding environment (Vocareum). Student should submit the homework using the online environment ONLY. Submissions of other kind will not be accepted. Late submission date and time are taken seriously. Be aware of your clock! Do not submit after the final deadline. Failure to follow instructions will result in penalties. Write the programs in the places provided. Homework 3 folder must contain the main program. Homework3-Extra-credit must contain the extra credit program.

Due Date: April 27th

- Late homework: you lose 20% of the homework's grade per 24-hour period that you are late. Beware, the penalty grows very fast: grade = points * (1 – n * 0.2) where n is the number of days late (n=0 if submitted on time, n=1 is submitted between 1 second and 24h late, etc).
- Grade review/adjustment: Requests will be considered up to 2 weeks after the grade is released. After that, it will be too late and requests for grading review will be denied.
- Homework assignments are to be solved individually.
- You are welcome to discuss class material in review groups, but do not discuss how to solve the homework.

Grading Policy

- Your source code must be written in Python.
- Your code must compile and execute on Vocareum with python 2.7.
- Your programs will be tested on test cases which would be different from the ones provided in input files.
- The source code must be properly commented.
- Your program should run in a reasonable time.
- You will use Vocareum.com to submit your code. Please check "Student Help.pdf" in the Homework-2 folder for instructions on how to use the system.

1. Overview [100 Points]:

Bayesian Networks, also called Belief or Causal Networks, are a part of probability theory and are important for reasoning in AI. They are a powerful tool for modeling decision-making under uncertainty. In this assignment you will implement code for computing inferences in Bayesian networks of discrete random variables.

You are a newly hired AI specialist at a large urban hospital. The hospital management has heard about the recent advances in AI technology and would like to see if it could be used in context of the hospital. In particular, management is interested in seeing AI approaches could be used to help doctors make more correct diagnoses.

The hospital already has a fairly extensive electronic medical record system, and they know for various diseases, what the priori probability of the diseases, P(D). In addition, they have also been able to determine the probability of various lab results, given one of the particular diseases P(F | D). You can assume that these probabilities are conditionally independent. The lab results, also called Findings, are binary random variables, that is, they are either present(true) or absent(false). Because some of the lab results are very expensive and others take a while to run, not all patients have all results available.

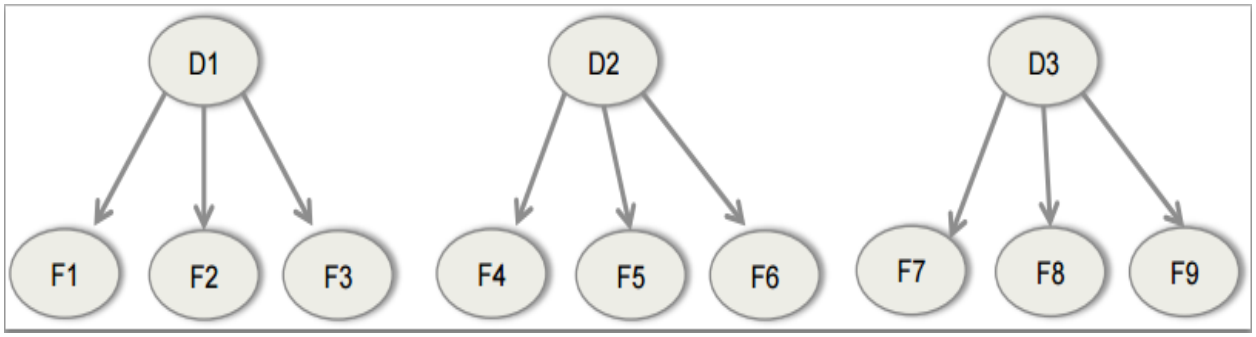

Figure 1: Bayesian network for diseases and findings

Symptoms are similar to findings, but they are things that a patient would report (such as "sore throat") rather than something that is determined by lab results (such as "hypokalemia"). Here, we treat findings and symptoms interchangeably.

Your job is to write a program that will accept as input the priori probabilities of a set of diseases and the conditional probability of a symptom (or finding) given each of the diseases.

2. Input:

The input file will contain the probabilities and inputs for Bayesian networks similar to shown in Figure 1. All the diseases have findings and symptoms attached with them. You can assume that the findings and symptoms for all the diseases are distinct.

The first line in the input file will have two numbers (n and k) separated by a white space specifying the number of diseases and number of patients respectively. Next 4*n lines will have the details about the diseases and their findings/symptoms, 4 lines for each disease. The first line for each disease will contain the name of the disease (Cancer, Hepatitis, etc), its number of findings/symptoms (m) and the priori probability of the disease P(D), all separated by a white space. The second line will contain a python list of m finding/symptom names for the disease. You can assume that these names are distinct. The third line will contain a python list of m elements giving the probability of the findings/symptoms to be present (true) if the disease is present. The forth line will contain a python list of m elements giving the probability of the findings/symptoms to be present(true) if the disease is not present.

The next n*k lines will have the inputs for various patients. For each patient, we have n lines representing the status of the findings for each disease. Each

line will contain the value of the available findings/symptoms for a disease, represented as a python list of values. The values can be T, F or U where T means that the finding for the patient is present(true), F means that the finding for the patient is absent(false) and U means that the finding for the patient is not available. You can assume that the values in this list correspond to the order of the findings listed in the disease description.

Example input:

```
2 2
diabetes 3 0.13
['thirst' , 'weightloss' , 'blurredvision']
[0.6, 0.7, 0.9]
[0.2, 0.3, 0.015]
Chickenpox 4 0.17
['fever' , 'sorethroat' , 'stomachache' , 'cough']
[0.7, 0.6, 0.3, 0.7]
[0.2, 0.1, 0.1, 0.2]
['T', 'F', 'U']
['T', 'U', 'F', 'U']
['T', 'F', 'T']
['F', 'U', 'T', 'T']
```

Figure 2: Example input

Explanation of the example:

- diabetes has 3 findings/symptoms: thirst, weightloss, blurredvision
- P(thirst | diabetes) = 0.6, P(weightloss | ¬diabetes) = 0.3
- There are inputs for 2 patients
- First patient has thirst, doesn't have weightloss and we don't know about the blurredvision

Notes:

- You can assume that the input file is without any error i.e. all the inputs provided are in the correct format.
- For each patient, for each disease, we have at least one test result available, either true of false.

- Like the previous assignment, you can use eval function of python to read the inputs which are in the list.
- In the final testing inputs, there will be at most 10 diseases and at most 10 findings/symptoms per disease.
- The command line for the program will be:

python bayes.py –i inputfilename

3. Output:

Your output file will contain the answers for the following questions for each patient. Your output should start with the patient number, for e.g. "Patient-1:" (start with 1). You should have 3 lines per patient. Each line is the answer to one of the questions below. If the input file name is "abc.txt" then the output file name for that input file must be "abc_inference.txt".

Question-1:

For each of the diseases, what is the probability that the patient has the disease? Note that a patient could have more than one disease at a time, so the probabilities of the diseases will not necessarily sum to 1. Also, because some of the lab tests are expensive, not all the patients will have all the lab results available, but your program should still produce correct results. The output will be a python dictionary with the name of the diseases as keys and a number specifying the probability of the patient having that disease as value.

- Present the values with 4 digit precision
- As you know, python dictionaries have no order. So output order of the elements is not fixed.

Question-2:

If not all the test results are available for a particular disease for a patient, your program should search the values for the unknown tests that would produce the *maximum* and *minimum* probabilities for each disease. This information could be used to help guide the patient, doctor (and insurance company) in deciding whether or not further tests are needed. The output will be a python dictionary with name of the diseases as keys and a list of 2 elements specifying the minimum and maximum probabilities for that disease as the value.

- These probabilities will be the same as Q-1 for diseases for which all the test results are available.
- Present the values with 4 digit precision.

Question-3:

In the case that there are undetermined lab values, to help the doctor decide which test to run next, the program should figure out which of the tests not done yet for each disease (result either true or false) would produce the biggest increase and which would produce the biggest decrease in the probabilities for that diseases. Your program should output a python dictionary with name of the diseases as keys and list of 4 elements as values. The first element in the list would produce the biggest increase in the probability of the disease. The second element is the value of the first element. The third element would result in the biggest decrease in the probability for the disease. And the forth element is the value of the third element.

- In case of ties, report the finding that comes earlier in the alphabetical order.

- If no findings would increase or decrease the probability of the disease or if all the findings for the disease are already available, report "none" as 1st and 3rd elements and 'N' as 2nd and 4th elements.

Example output:

```
Patient-1:
{'Chickenpox': '0.3580', 'diabetes': '0.1612'}
{'Chickenpox': ['0.0850', '0.9213'], 'diabetes': ['0.0191', '0.9202']}
{'Chickenpox': ['sorethroat', 'T', 'cough', 'F'], 'diabetes': ['blurredvision', 'T', 'blurredvision', 'F']}
Patient-2:
{'Chickenpox': '0.4464', 'diabetes': '0.9202'}
{'Chickenpox': ['0.2639', '0.8287'], 'diabetes': ['0.9202', '0.9202']}
{'Chickenpox': ['sorethroat', 'T', 'sorethroat', 'F'], 'diabetes': ['none', 'N', 'none', 'N']}
```

Figure 3: Example output

Explanation of the example:

- The first patient has the chickenpox with the probability of 0.3580 and diabetes with the probability of 0.1612.
- If all the unknown symptoms were known in a way that minimized the probability of chickenpox, the minimum probability of the first patient having chickenpox will reduce to 0.0850 and correspondingly if all the unknowns were known with values that maximized the disease probability, the maximum probability will be 0.9213.
- In case of chickenpox for the first patient, sorethroat and cough are unknown. Having sorethroat = True, will produce the biggest increase in the probability that this patient has chickenpox (resulting in symptoms = ['T', 'T', 'F', 'U']). If we have cough = False, it will result in the biggest decrease in the probability of him having chickenpox (resulting in symptoms = ['T', 'U', 'F', 'F']).

4. Extra Credit [20 Points]:

Now let's get a bit more realistic over here. If you go and ask around professionals about these prior probabilities and conditional probabilities, you will get a variety of answers from them. So what is the solution to that? The best way to do this is to get the actual data and then create the CPTs

(Conditional Probability Tables) from those rather than relying on someone else's inputs and be hopeful that it would be correct!!

For this part, you have now changed your job and now you are an analyst at an insurance company. The company has a lot of data which they would like to use to enhance/modify their policies. The company asked a lot of professionals regarding what are the risk factors that cause certain diseases and they collected the data based on that. You are given a Bayesian network of these factors as a directed graph indicating the causal effect. The graph is shown in Figure 4.

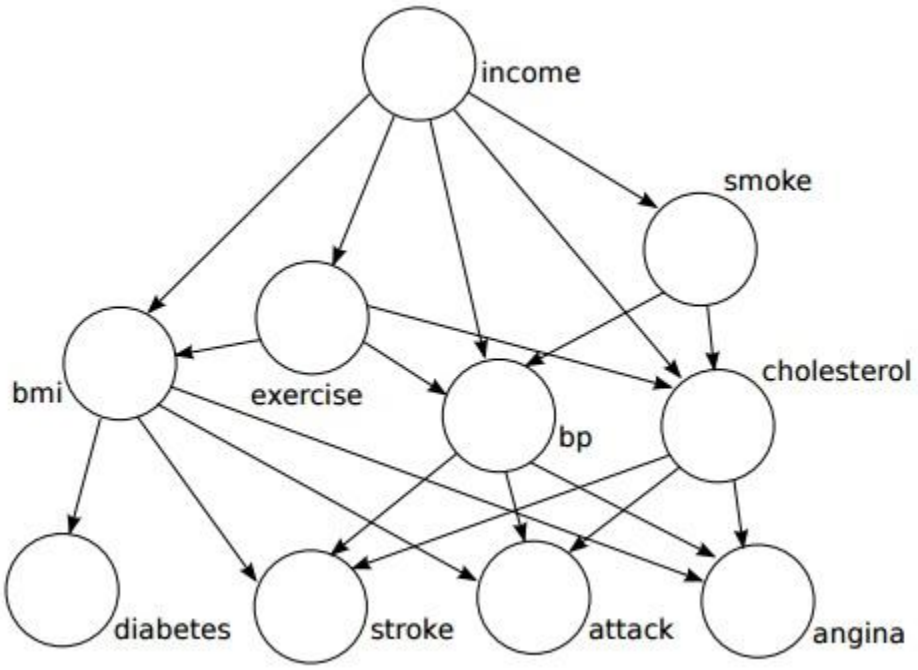

Figure 4: Risk Factors

You will be given a data file as one of the input files which will contain a record of 10000 of their customers that you will be using to answer the queries. The structure of the file is as follows:

- The first line contains the name of the fields in order which will be the same as the data. All the fields are separated by tab.

- The remaining 10000 lines contain values for the respective fields in the first line. All the values are separated by tab.

Here are the categories/possible values for each field:

- Income (income of the customer) – [<25000, 25001-50000, 50001-75000, >75000]
- Exercise (whether the customer exercises) – [yes, no]
- Smoke (whether the customer smokes) – [yes, no]
- BMI (what is the BMI of the customer) – [underweight, normal, overweight, obese]
- BP (whether the customer's BP remains high) – [yes, no]
- Cholesterol (whether the customer has high cholesterol) – [yes, no]
- Angina (whether the customer has angina problem) – [yes, no]
- Attack (whether the customer has heart attack problem) – [yes, no]
- Stroke (whether the customer has stroke problem) – [yes, no]
- Diabetes (whether the customer has diabetes problem) – [yes, no]

You need to create a program, ***riskFactor.py*** that will read the data file provided to create the Conditional Probability Tables (CPTs) for each of the node in the graph. You do not need to report it in the output so you can choose any structure to store you CPTs and access them to answer the queries.

INPUT & OUTPUT:

The input to your program will be a file containing queries which you will answer using the CPTs you will infer using the input data file. An example query would be "what is P(diabetes=yes | exercise=yes, smoke=no)?". For each query, you need to output the probability. The input file to the program will contain a number n in the first line which is the number of queries in the

input file. The next n lines in the file will contain one query per line. Here is the format of the queries:

[{'diabetes':'yes'}, {'exercise': 'yes', 'smoke': 'no'}]

Each query will be a python list. The first element in the list which is a dictionary will describe the query nodes ('diabetes' in the example shown). So the symptoms that we are looking for are the keys and their values are the dictionary values. The second element in the list will be a python dictionary containing the evidences for the query. For each element in the dictionary, the key will be evidence node and the value will be the value for that evidence node ('exercise' and 'smoke' are the evidence nodes in the example shown for which the values are 'yes' and 'no' respectively). You will need to output the probability of the query nodes based on the evidences provided in the query for each query in the input file. Your output file, ***riskFactor.out*** will contain n lines, one for each input query.

Notes:

- The command line for the program will be:

Python riskFactor.py –i inputfilename –d datafile

- For the given Bayesian network you can assume the conditional independence relationship as "a node is independent of its ancestors given its parents".
- You are given a sample data file with the homework. The test data file will be different from the one give but will have the same format and for the same graph. So your program needs to figure out the values of the CPTs on the fly. Keep in mind that the structure of the CPTs will remain the same.
- The first dictionary of the list is always non-empty. But the second one can be empty. i.e. [{'diabetes':'yes'} , {}]

- You can assume that there is no conflict between the query and the evidences. So you do not have something like:

  [{'diabetes':'yes'}, {'diabetes'|'no'}]

- In class on April 9th we will talk about using "add-one smoothing" versus the maximum likelihood estimate for estimating a probability based on data. For this problem, use the maximum likelihood estimate, do not use add-one smoothing.
- You can assume that the data given will be valid, i.e. all the values for each of the fields will be one of the valid values for that field.
- Present the values of probabilities with 4 digit precision in your output.