```cpp
// <NaiveBayes rollNo=8390 name='Joyson' />

#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <iterator>
#include <map>

using namespace std;

auto countFreq(vector<string> vect, int n) {
    map<string, float> mp;

    for(auto x : vect)
        mp[x]++;

    return mp;
}

int main() {
    int n,m,i,j,e;
    string temp,temp1,temp2;
    cout<<"Enter number of tuples\n";
    cin>>n;
    cout<<"Enter number of attributes\n";
    cin>>m;

    vector<string> b, newtuple, classes, raw_list;
    vector<float> probabilities;
    map<string, float> frequency, prob_match;
    vector<vector<string>> a;
    a.resize(n);
    for(i=0; i<a.size(); i++)
        a[i].resize(m);

    cout<<"Enter attribute labels (Restrict label names within 7 alphabets)\n";
    for(i=0;i<m;i++) {
        cin>>temp;
        b.push_back(temp);
    }


    for(i=0;i<n;i++) {
        cout<<"Enter tuple "<<i+1<<" (Restrict attribute values within 7 alphabets)
        \n";
        for(j=0;j<m;j++) {
            cin>>temp;
            a[i][j]=temp;
        }
    }

    cout<<"\n\n||\t\t||";
    for(i=0;i<m;i++) {
        cout<<"\t"<<b[i]<<"\t||";
    }
    cout<<"\n";
    for(i=0;i<20*m;i++)
        cout<<"_";
    cout<<"\n";

    for(i=0;i<n;i++) {
        cout<<"||\t"<<i+1<<"\t||";
        for(j=0;j<m;j++) {
```

```
64              cout<<"\t"<<a[i][j]<<"\t||";
65          }
66          cout<<"\n";
67      }
68
69      cout<<"\n\nEnter new tuple to be classified (Restrict attribute values within 7
        alphabets) \n";
70      for(i=0;i<m-1;i++) {
71          cin>>temp;
72          newtuple.push_back(temp);
73      }
74
75      for(i=0;i<n;i++) {
76          temp = a[i][n-1];
77          bool present = binary_search(classes.begin(), classes.end(), temp);
78          if(!present)
79              classes.push_back(temp);
80      }
81
82      for(i=0;i<n;i++) {
83          for(j=0;j<m;j++) {
84              raw_list.push_back(a[i][j]);
85          }
86      }
87
88      e = sizeof(raw_list)/sizeof(raw_list[0]);
89      frequency = countFreq(raw_list, e);
90
91      for(auto x : classes) {
92          float prod = 1;
93          for(auto y : newtuple) {
94              float count = 0;
95              for( auto z : a) {
96                  bool contains1 = 0, contains2 = 0;
97                  if(std::find(z.begin(), z.end(), x)!=z.end())
98                      contains1 = 1;
99                  if(std::find(z.begin(), z.end(), y)!=z.end())
100                     contains2 = 1;
101                 if(contains1 && contains2)
102                 {
103                     count++;
104                 }
105             }
106             prod = prod * ((float)count/(float)frequency[x]);
107         }
108         prod = prod * ((float)frequency[x]/(float)n);
109         prob_match[x]=prod;
110     }
111
112     float sum = 0;
113     for (auto x : prob_match)
114         sum+=x.second;
115
116     for (auto &x : prob_match)
117         x.second = x.second/sum;
118
119     for(auto x : prob_match)
120         cout<<"\n"<<x.first<<" : "<<x.second*100<<"%"<<endl;
121
122     return 0;
123 }
124
125
126
```

```
/*
Enter number of
tuples
5

Enter number of
attributes
5

Enter attribute labels (Restrict label names within 7
alphabets)
Outlook Temp Humidty Windy
Play
Enter tuple 1 (Restrict attribute values within 7
alphabets)
Rainy Hot High False
No
Enter tuple 2 (Restrict attribute values within 7
alphabets)
Rainy Hot High True
No
Enter tuple 3 (Restrict attribute values within 7
alphabets)
Ovrcast Hot High False
Yes
Enter tuple 4 (Restrict attribute values within 7
alphabets)
Sunny Mild High False
Yes
Enter tuple 5 (Restrict attribute values within 7
alphabets)
Sunny Cool Normal False
Yes



||      ||  Outlook ||  Temp  ||  Humidty ||  Windy  ||  Play  ||
_____________________________________________________________
||  1   ||  Rainy   ||  Hot   ||  High    ||  False  ||  No    ||
||  2   ||  Rainy   ||  Hot   ||  High    ||  True   ||  No    ||
||  3   ||  Ovrcast ||  Hot   ||  High    ||  False  ||  Yes   ||
||  4   ||  Sunny   ||  Mild  ||  High    ||  False  ||  Yes   ||
||  5   ||  Sunny   ||  Cool  ||  Normal  ||  False  ||  Yes   ||



Enter new tuple to be classified (Restrict attribute values within 7
alphabets)
Rainy Hot High
True


No :
100%



Yes : 0%
*/
```