DSA/CS 5005 – Computing Structures – Programming Project – 2 – Summer 2016
(Due: July 18, 2016 11:59 PM)

Objectives: Learn to build and manipulate trees. You will build and manipulate Attribute Trees that are k-ary trees. You will learn to use the list STL and vector data structures along with vector and the String class.

Project Description: You input consists a table of data (all of which are String data types, except the last elements in the last column which is an int data type). Each column of the table is referred to as an attribute. The number of columns minus one of the table is called the dimensions of the table. In each row of the table the last column corresponds to the count of the number of records (in some database, which we do not have to consider in this project) that are exactly same as the row (all column values are the same). Consider the following table. This table 16 rows and 5 columns. The last column corresponds to the count value. For example, consider row 1. The number of records in some data base with values for Attribute1-4 as Afghanistan, Low, Dense, 100% is 10.

No	Attribute1	Attribute2	Attribute3	Attribute4	Count
1	Afghanistan	High	Dense	50%	10
2	Afghanistan	High	Sparse	50%	30
3	Afghanistan	Low	Dense	100%	10
4	Afghanistan	Low	Dense	50%	20
5	Algeria	High	Dense	50%	20
6	Algeria	High	Dense	100%	10
7	Algeria	Low	Dense	50%	20
8	Algeria	Low	Dense	100%	20
9	Belarus	High	Average	50%	10
10	Belarus	High	Average	100%	10
11	Cameroon	High	Dense	50%	30
12	Cameroon	Medium	Average	100%	40
13	Chile	Low	Sparse	100%	50
14	Chile	Medium	Sparse	50%	30
15	Chile	Medium	Dense	50%	40
16	Chile	Medium	Dense	100%	30

(a) The first line of the input will consists of two integers (noAttributes, noRows). After you read these two values, you are supposed to create an array of attributeValue objects (size of this array is noRows), where each object stores all the values for each of the attributes in that row and the count value.

(b) Next you will create an array of unique attribute values for each of the attributes (call it uniqueAttrValues. This will be an array of vectors (all storing String data types), where the size of the array is equal to the number of attributes (the first location of the array will correspond to the first attribute and so on). Each vector will store the unique values for that attribute. See the following example for the above table.

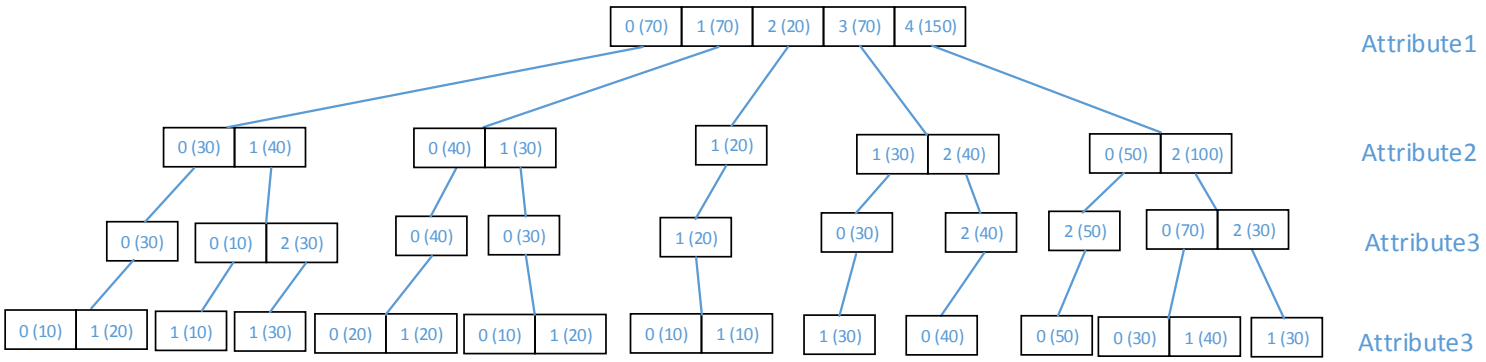
	0	1	2	3	4
0	Afghanistan	Algeria	Belarus	Cameroon	Chile
1	Low	High	Medium		
2	Dense	Average	Sparse		
3	100%	50%			

(c) A matrix named tableValues is created next (Rows = noRows, Cols = noAttributes+1). The tableValues is a matrix (of all ints) that tables the tabular data shown earlier and replaces the attribute values in each row with the vector position on the uniqueAttrValues data structure. For example, for the tabular data above the tableValues matrix is shown below.

No	Attribute1	Attribute2	Attribute3	Attribute4	Count
1	0	1	0	1	10
2	0	1	2	1	30
3	0	0	0	0	10
4	0	0	0	1	20
5	1	1	0	1	20
6	1	1	0	0	10
7	1	0	0	1	20
8	1	0	0	0	20
9	2	1	1	1	10
10	2	1	1	0	10
11	3	1	0	1	30
12	3	2	2	0	40
13	4	0	2	0	50
14	4	2	2	1	30
15	4	2	0	1	40
16	4	2	0	0	30

(d) Using the tableValues matrix we will build our Attribute Tree (AT) data structure. The AT data structure can now be constructed with above table and it is shown below. Please note that the numbers in the parenthesis are the count values. For example on the first level of the tree shown below we see 0 (70). The 0 is the unique value of Attribute1 and count is the number of records with Attribute1 equal to 0; in this case it is 70 (look at the table above).

As part of project, you are required to build the Attribute Tree and each node of the attribute tree could capture the value of count. One way to create the tree is as follows. Each node of the tree has three vectors: one for storing the values, the count for each value, and pointers to nodes of the tree. Write an algorithm to insert row by row into the tree and moving down the tree after you insert each value into the corresponding vector.



After you build the tree (with the initial data) you will read a number (int) that tells you the number of command lines you need to read from the input file. Each command line will either start an F (for Find the count). Here are some examples:

F * * * *	//Give the total number of records the above example it is 380
F Afghanistan * * *	//The output is 70
F Cameroon High * *	//The output is 30
F Chile * Sparse *	//The output is 80

Point Distribution:

(a) Creating the arrtibuteValues class and the array to read in the input (15%)
(b) Creating the uniqueAttributes data structure (15%)
(c) Creating the tableValues matrix (15%)
(d) Constructing the Attribute Tree (classes and methods) (35%)
(e) Processing the find queries (20%)
(f) Bonus (20%) Perform insert operation (use I in the command) to insert rows into the Attribute Tree. For example, I Afghanistan Low Sparse 50% 50 will insert new nodes into the tree and update the count values in the intermediate nodes.

Constraints:

1. This project must be implemented in C++. The headers you can use are <iostream>, <vector>, <fstream> and <string>. We will not use any other libraries.
2. None of the projects in DSA 5005 will be a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.
3. The project is due on July 18, 2016 by 11:59pm. Source file(s) should be submitted to the digital drop box. Multiple source files must be submitted in a single zip file.

Late Penalty:

Submit your project on or before the due date to avoid any late penalty. A late penalty of 10% per day will be imposed after the due date. After five days from the due date, you will not be allowed to submit the project. If your submitted project doesn't work (or compile), you will lose 50% of the project grade.