Program Set #4
Total Points: 30

Three problems must be implemented for full credit. Each section will state how many problems to implement within it. The required (starred) problem marked in the set must be implemented by everyone. See 2436 Grading Guide Sheet for additional grading/submission information. Partial credit will be given. (10 points each)

Section One- Binary Trees/General Trees. Choose one problem.

1. Write a program that takes any word, like `INVITATIONAL`, and builds a binary search tree in alpha order, allowing duplicate letters, which will be inserted to the left when the duplicate is encountered, and then calculates five traversals: in-order, pre-order, post order, level order, and reverse order. A reverse order traversal is just the in-order traversal backwards. Input will be a one-word string (all caps) greater than 3 letters long. Assume proper input. The program will output to the screen the original string, the string using each the 5 traversals stated above (labeled) on separate lines, and the tree printed in vertical form.

The vertical tree should consist of a line of nodes, followed by a line of / and \ characters indicating relationships, followed by a line of nodes, etc. Assume all nodes are representable as a single character. Adjacent nodes on the lowest level should be separated by at least one space, nodes further up should be separated as appropriate. Nodes with two children should be placed precisely in the middle of their direct children. Relationship slashes should be halfway between the parent and the appropriate child (round whichever way you want). Must use the Java/C++ string classes for string input. A tree data structure must be used. Refer to the sample output below.

Sample Run:

```
Enter a word: JAMES

JAMES

Preorder traversal:             J A E M S
Post order traversal:           E A S M J
In order traversal:             A E J M S
Level order traversal:          J A M E S
Reverse order traversal:        S M J E A

Reconstructed Tree:

       J
     /   \
    A     M
     \     \
      E     S

Enter a word: INVITATIONAL
```

```
INVITATIONAL

Preorder traversal:            I I A A I N N L V T T O
Post order traversal:          A I A I L N O T T V N I
In order traversal:            A A I I I L N N O T T V
Level order traversal:         I I N A N V A I L T T O
Reverse order traversal:       V T T O N N L I I I A A

Reconstructed Tree:

        I
      /   \
    I       N
   /       /  \
  A       N   V
 / \     /   /
A   I   L   T
           /
          T
         /
        O
```

Name the program: `TreeTraversalsXX.java` or `TreeTraversalsXX.cpp`, where XX are your initials.

2. Write a program given a list of pairs of outline topics, each pair being a subtopic and the topic it belongs under and outputs a nicely formatted outline. The input file will contain data for multiple outlines that need to be constructed. The data for each outline starts with a line containing only a positive integer, `n (n < 300)`. The next `n` lines each contain exactly one subtopic and the topic it belongs under, in the form:

```
subtopic/topic
```

The subtopic and topic will both start with a letter, and both be limited to 40 characters. Each may contain any characters other than the "/" separator, but will not contain more than one consecutive space, nor end with a space. Note that the subtopic on one line of the input might appear—identically, character for character, with letters case-sensitive—as a topic in one or more other input lines for the same outline, since it might have subtopics of its own. Each topic for an outline is guaranteed to appear in that final outline exactly once. The data for the last outline will be followed by a line containing only a zero.

For each outline, output to the screen the message "`Outline #m`" where `m` is the number of the outline from the input, starting at 1. Beginning on the next line, output the outline. Show all the top-level topics (the ones that aren't subtopics of any other topics) in sorted ASCII order (where capital Z comes before lowercase a), with each on its own line, and starting at the beginning of the line. Starting on the line immediately after each topic, show its subtopics, also ASCII sorted, on separate lines and indented 4 spaces from where that topic starts. For example, a subtopic of a subtopic of a top-level topic will be indented 8 spaces, and its subtopics, in turn, will be indented 12 spaces. Output a blank line after each outline. Let the user input the file name from the keyboard. Use a tree data structure. Refer to the sample output below.

Sample File:

```
16
Characters/Anathem
WW2/Cryptonomicon
sphere/avout
nanotechnology/Diamond Age
Erasmas/Characters
bolt/avout
Jad/Characters
avout/Vocabulary
TL;DR/Baroque Cycle (3 books)
concent/Vocabulary
Cord/Characters
cord/avout
Arsibalt/Characters
dot-com/Cryptonomicon
Neo-Victorian/Diamond Age
Vocabulary/Anathem
5
Betty/Rubble
Fred/Flintstone
BAMM-BAMM/Rubble
Wilma/Flintstone
Barney/Rubble
0
```

Sample Run:

```
Enter file name: outline.txt

Outline #1
Anathem
    Characters
        Arsibalt
        Cord
        Erasmas
        Jad
    Vocabulary
        avout
            bolt
            cord
            sphere
        concent
Baroque Cycle (3 books)
    TL;DR
Cryptonomicon
    WW2
    dot-com
Diamond Age
    Neo-Victorian
    Nanotechnology

Outline #2
Flintstone
    Fred
    Wilma
Rubble
    BAMM-BAMM
    Barney
    Betty
```

Name the program: `OutlineXX.java` or `OutlineXX.cpp`, where `XX` are your initials.

Section Two- Graphs. Choose one problem.

3. Your neighbor is building a social networking site and is implementing the feature for referring a friend. When someone is using the site, they need to refer to them the best candidate for friendship. This is defined as a friend of the current user's friends who has the most friends in common with the user. For example, Anne has two friends Beth and Julian. Beth has three friends Anne, Julian, and Chris. Julian has three friends Anne, Burton, and Chris. Burton has one friend, Julian, and Chris has two friends,

Beth and Julian. For Anne, there are two friends of friends to consider: Burton and Chris. In this example Anne shares the greatest number of friends with Chris, (Beth and Julian) where Burton only shares one friend (Julian), therefore Chris is the best candidate for friendship with Anne. For Beth, only one friend of her friends, Burton, is not already her friend, therefore, he is the best and only candidate for friendship.

Write a program given a person and their list of friends output the best new candidate for friendship. Input will be from a data file with several sets of data, each consisting of a list of persons with their respective list of friends followed by a `"-"`. Each user's name will be followed by a `":"` followed by the friends of that user. After the `"-"` will be a list of users by name, each on a separate line. The end of each complete data set is indicated by a `"*"`. For each labeled data set, output to the screen the username followed by a colon followed by the name of the best candidate for a new friend. Assume the friends circle graph is closed, and it is guaranteed that there will only be one friend to be considered as the best candidate for friendship. Let the user input the file name from the keyboard. Use a graph data structure. Refer to the sample output below.

Sample File:
```
Anne: Beth Julian
Beth: Anne Julian Chris
Julian: Anne Burton Chris
Chris: Beth Julian
Burton: Julian
-
Anne
Beth
*
Aaron: Jenna Fallon Aftub
Jenna: Fallon Darius Aaron
Fallon: Aaron Jenna Aftub Darius
Darius: Jenna Fallon
Aftub: Aaron Fallon
-
Aaron
Darius
Jenna
*
```

Sample Run:
```
Enter file name: people.txt

Data Set 1:
Anne: Chris
Beth: Burton

Data Set 2:
Aaron: Darius
Darius: Aaron
Jenna: Aftub
```

Name the program: `FriendCandidatesXX.java` or `FriendCandidatesXX.cpp`, where `XX` are your initials.

4. Consider a field 20 x 20 containing random numbers, 0-4. It represents an unevenly leveled terrain to travel through. It takes 1 hour to travel from one square to the next, provided the two squares have the same height. If the difference is 1, then it takes 2 hours; if the difference is 2 it takes 4 hours; if the difference is 3 it will take 6 hours; and if the difference is 4 it will take 8 hours. Travel may only take place from a square to a square directly north, south, east, or west of it (no diagonals). Write a program to find shortest time to get from the upper left corner to the diagonally opposite corner, i.e., from `(1,1)` to `(20,20)`. Input from a text file, where the first integer `N` will represent the number of data sets. Each

data set contains 20 lines of 20 single digits each from 0 to 4 inclusive representing the 20 x 20 field to travel through. Output to the screen each data set as follows:

```
Field X: Y hours
```

where X is the data set number and Y is the total hours it takes to travel. A graph data structure must be used. Let the user input the file name from the keyboard. Refer to the sample output below.

Sample File:

```
2
10211022313443041012
13020104041032141110
11111111133230001441
24131014140241121032
04013240103021231110
34141400110113404232
41301104120243003314
41320023114042124314
03212124133310301021
43401131111111004304
40213310024241210420
04414142130101244021
33332323301431323211
22114313443131230400
01431031214341223104
43013144212231111111
33213001012124104321
22044443324433333221
14411202232212411021
14323112402331210201
32421201102201034141
01012402302032140101
43430140031143420441
40142243030212321001
04412300331314202323
02011441434344030343
44133340433310134112
23401403031302132224
12413113241122430112
34133214030013002110
33423312414304233022
41410213122430122034
04042042431141131331
31302111314034111214
40322221024131104442
04311213013242402014
44042431204112420214
32242223400244433013
41243444044124414410
42311114101201213330
```

Sample Run:

```
Enter file name: numpath.txt

Field 1: 38 hours
Field 2: 76 hours
```

Name the program: `NumberSPXX.java` or `NumberSPXX.cpp`, where `XX`are your initials.

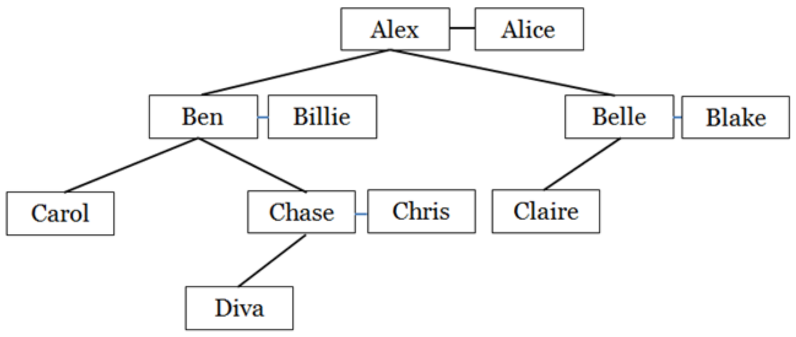
Required Problem- Comprehensive.

5 (**). Your aunt has been taking an interest in the genealogy of the family. She can construct a family tree but is getting confused with the relationships between different members of the family. She wants to identify the following relationships: father, mother, uncle, aunt, son, daughter, nephew, niece, cousin, husband and wife. She also wants to be able to recognize whether the members in the family are related by blood or by marriage (i.e. in-laws), as well as different generations in the family such as grandparents, grandchildren, great grandparents, great grandchildren, great great grandparents, great great grandchildren etc. and different degrees of cousins including levels of removedness (e.g. second cousins-in-law twice removed). Write a program to construct the family tree and name the relationships. Here are the following conditions:

- No second marriages which require step relationships e.g. stepbrother and stepfather will be recorded
- All children in the family tree will be the offspring of a male father and a female mother who are married
- No marriages between siblings or between cousins of any type have occurred
- All people in the family tree may not be related.

Further to those conditions, the following definitions apply:

- Father and mother are the parents of a child
- Brother and sister are male and female siblings with the same parents
- Son and daughter are the children of a parent
- Uncle and aunt are the brother and sister of a child's parent
- Nephew and niece are the male and female children of a sibling
- A grandfather and grandmother are the male and female parents, respectively of a child's parent
- A great grandfather and great grandmother are the father and mother, respectively, of a child's grandparent
- A great uncle or great aunt is a sibling to a child's grandparent
- Cousins (not removed) are at the same level in the family tree:
    - first cousins have the same grandparents
    - second cousins have the same great grandparents
    - and so, on
- Removed cousins are at different levels in the family tree:
    - a first cousin once removed is the child of one of the first cousins
    - a first cousin twice removed is the grandchild of one of the first cousins
    - and so, on
- Cousin relationships are symmetric e.g. if A is the first cousin twice removed of B, B is also the first cousin twice removed of A
- Where a relationship occurs due to marriage of two people the relationship is said to be in-law:
    - the parent of a person's husband or wife is a father-in-law or mother-in-law
    - the sibling of a person's husband or wife is a brother-in-law or sister-in-law
    - the cousin of a person's husband or wife is a cousin-in-law



The picture displays the family tree described in the sample output. The program should be able to say that Claire and Carol are 1st cousins, Claire and Diva are 1st cousins 1-time removed, and Claire and Chris are cousins-in-law. The program should also be able to generate any other relationship combinations when queried.

The input from a data file consists of a list of relationships for construction of the family tree. The list of relationships will be followed by a list of queries for which you will name the relationship. Relationships will be provided to infer the gender of all family members. All relationships will be in lower case and all names will be unique. At most one person in each marriage will have parents present in the input. The first line of datafile input contains a single integer `r` (`1 ≤ r ≤ 200`) being the number of relationships for building the family tree. `r` lines of relationship definitions follow. Each relationship consists of three alphabetic strings, `name1`, `name2` and `relation`, each separated by a single space. relation will be one of husband, wife, son or daughter. The relationship line can be read as:

```
name1 is the relation of name2
```

The relationships are followed by a line containing a single integer `q` (`1 ≤ q ≤ 200`) being the number of queries on the family tree. `q` query lines follow. Each query line consists of two strings, `name1` and `name2` separated by a single space. The names in the queries do not necessarily appear in the relationship pairs.

For each relationship query, output to the screen the relationship between `name1` and `name2` on a single line. In the following definitions mandatory items are delimited with `(` and `)`, optional items are delimited with `[` and `]`, options are separated by `|`. Elements which may require repetition (1 to many) are followed by `*`.

- For a spousal relationship i.e. husband or wife, output a sentence of the following form:

  `name1` is the `(husband|wife)` of `name2`

- For a sibling relationship i.e. brother or sister, output a sentence of the following form:

  ```
  name1 is the (brother|sister)[-in-law] of name2
  ```

- If the relationship is some kind of cousin, output a sentence which includes the degree of cousinship i.e. `1st, 2nd, 3rd` etc. followed by the word cousins, then the suffix -in-law if and only if the relationship is by marriage and finally the number of times removed (1-time removed, 2-times removed, 3-times removed, etc.).

```
name1 and name2 are (1st|2nd|3rd|...) cousins[-in-law][ (1-time|2-times|3-times|...) removed]
```

- If the relationship is aunt, uncle, nephew or niece, the output may require one or more instances of the word great.

  ```
  name1 is the [great ]*(aunt|uncle|nephew|niece)[-in-law] of name2
  ```

- Otherwise the relationship will be one of son, daughter, father or mother. Relationships which are two generations apart will require the use of the word grand before the relationship. Relationships which are more than two generations apart will require the use of one or more instances of the word great before the word grand.

  ```
  name1 is the [[great ]*grand](son|daughter|father|mother)[-in-law] of name2
  ```

- There is `NO RELATION` if either of the names in the queries do not appear in the list of relationships.

Let the user input the file name from the keyboard. You must use a tree data structure. Refer to the sample output below.

Sample File

```
10
Alex Alice husband
Ben Alex son
Chase Ben son
Diva Chase daughter
Carol Ben daughter
Belle Alex daughter
Chris Chase wife
Blake Belle husband
Billie Ben wife
Claire Belle daughter
10
Claire Carol
Claire Diva
Claire Chris
Billie Belle
Billie Chris
Billie Chase
Belle Carol
Blake Carol
Carol Belle
Carol Blake
```

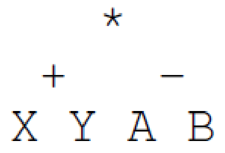
Sample Run:

```
Enter file name: family.txt

Claire and Carol are 1st cousins
Claire and Diva are 1st cousins 1-time removed
Claire and Chris are 1st cousins-in-law
Billie is the sister-in-law of Belle
Billie is the mother-in-law of Chris
Billie is the mother of Chase
Belle is the aunt of Carol
Blake is the uncle-in-law of Carol
Carol is the niece of Belle
Carol is the niece-in-law of Blake
```

Name the program: `FamilyRelationsXX.java` or `FamilyRelationsXX.cpp`, where `XX` are your initials.

Extra Credit: Choose one of the two problems below for extra credit. See 2436 Grading Guide Sheet for additional grading/submission information. Partial credit will be given. No additional extra credit for implementing both programs.



1. The expression `(X + Y) * (A - B)` could be represented in a binary tree as illustrated to the side. The root node of the tree contains the operator (*) that is evaluated last. Each subtree contains an operator and itself a complete expression. The subtree to the left of the root node represents the expression `(X+Y)`; and the subtree to the right of the root node contains the expression `(A-B)`. Write a program to output a binary expression tree in prefix and postfix form and output the expression in tree form horizontally to the screen. Input from the keyboard an expression that can contain parentheses (to any level of nesting), `+, -, *, /`, and letters (all caps). There are no spaces in the expressions. Each expression is no more than 20 characters long. There are sufficient parentheses in each expression to indicate explicitly the order of operations. (the number of pairs of parentheses is one less than the number of arithmetic operators).

Output to the screen the original expression, its prefix and postfix forms with exactly one blank before and after a bracket, and one blank before and after an arithmetic operator on separate lines. Below the expressions, display the expression in horizontal (left to right) tree form where the top of the tree is at the left of the screen. Use an arrow "`->`" to indicate the branches. Use a minimum of one space between characters within the tree. Use a tree data structure. Refer to the sample output below.

Sample Runs:

```
Enter the expression: (A+B)/C

Expression: (A + B) / C
Prefix form: / + A B C
Postfix form: A B + C /
Expression Tree:

     -> C

 -> /
         -> B
     -> +
         -> A
```

```
Enter the expression: (X+Y)/(C*M)

Expression: (X + Y) / (C * M)
Prefix form: / + X Y * C M
Postfix form: X Y + C M * /
Expression Tree:

        -> M
    -> *
        -> C
-> /
        -> Y
    -> +
        -> X
```

Name the program: `ExpressionTreeXX.java` or `ExpressionTreeXX.cpp`, where XX are your initials.

2. Your aunt is part of a bike sharing program in her town. However, she has noticed recently that some of the bike stops in her town will have an abundance of bikes, whereas other stops will be completely empty. To help address where and how the bikes are moving, she went to every location at the beginning of the day and tagged one bike. She then monitored all the places the bike traveled that day. She needs your help to identify where the sources and sinks are in her bike share community stops. A

source is defined as a stop that one or more bikes left from and none came to. A sink is defined as a spot that one or more bikes came to, but none left from. All the stops are labeled with letters so your aunt can keep up with them.

Input from a data file where the first integer is the number of data sets to follow. Each data set will begin with an integer to represent the number of stops your aunt is monitoring. Each line will represent the bike that your aunt tagged, where it started, a "->" symbol, and then all places the bike traveled to (in no particular order). If a bike was not checked out, this will be noted as "NONE". Output to the screen, for each labeled data set, the list of alphabetized sources, prefaced by the word "Sources: " and the list of alphabetized sinks, prefaced by the word "Sinks: ". If there are none, then say "NONE". Let the user input the file name from the keyboard. Use a graph data structure. Refer to the sample output below.

Sample File

```
3
3
A -> A B C
B -> B
C -> NONE
4
E -> E
F -> F
G -> G
D -> NONE
5
Q -> B C V
B -> B C V
C -> B C
D -> B C V
V -> NONE
```

Sample Run:

```
Enter file name: sources.txt

Data Set 1:
Sources: NONE
Sinks: C

Data Set 2:
Sources: NONE
Sinks: NONE

Data Set 3:
Sources: D Q
Sinks: V
```

Name the program: GraphStopsXX.java or GraphStopsXX.cpp, where XXare your initials.