CS/SSW 555 Agile Methods for Software Development
GEDCOM Project Overview

During this course, you will practice some of the Extreme Programming (XP) and Scrum methods on a very small team project. You will develop a command-line program, analogous to lint and PyLint, to discover errors and anomalies in GEDCOM genealogy files.

You will be part of a small team with 3 or 4 members writing the application in Python, Java, or C/C++ and using an SQL or MongoDB database to store and query the ancestry information. Your team will choose the programming language and database solution that works for your team.

GEDCOM is a standard format for genealogy data developed by The Church of Jesus Christ of Latter-day Saints. GEDCOM identifies two major entities: individuals and families. GEDCOM allows you to describe the following characteristics of individuals:

- Unique individual ID
- Name
- Sex/Gender
- Birth date
- Death date
- Unique Family ID where the individual is a child
- Unique Family ID where the individual is a spouse

Likewise, GEDCOM allows you to describe the following characteristics of a family:

- Unique family ID
- Unique individual ID of husband
- Unique individual ID of wife
- Unique individual ID of each child in the family
- Marriage date
- Divorce date, if appropriate

GEDCOM is a line-oriented text file format where each line has three parts separated by blank space:

1. **level number** (0, 1, or 2)
2. **tag** (a string of 3 or 4 characters)

3. **arguments** (an optional character string)

All lines (or records) begin with a level number that is used to group the information from multiple records. Records with level numbers 1 and 2 are always in the form:

```
<level_number> <tag> <arguments>
```

Lines with level number `0` have one of two different forms. The first has the form

```
0 <id> <tag>
```

where *<tag>* is `INDI` or `FAM`. The *<id>* field between the "0" and the tag is a unique identifier used to identify an individual or a family.

The second version of level 0 records has the form

```
0 <tag> <arguments that may be ignored>
```

where *<tag>* is `HEAD`, `TRLR`, or `NOTE`.

For example,

```
0 p1 INDI
```

begins the definition of a new person with id 'p1'. The definition may include any number of records of level 1 or 2. The definition of person 'p1' ends either at the end of the GEDCOM file, or with the next level 0 record.

```
0 f207 FAM
```

begins the definition of a new family with id 'f207'.

```
0 NOTE this is comment
```

defines a new comment.

Level numbers provide a mechanism for grouping records in hierarchical groups: all records with larger level numbers than the current record belong to the entity headed by that entity. For example, all of the records from the first level 0 record, up to, but not including the "0 F32 FAM" record, belong to the group headed by the first record (with level 0) that describes the individual with ID "I27". The DATE record with level number 2 is associated with the previous BIRT record with level number 1.

```
0 I27 INDI
1 NAME Mark /Ardis/
1 SEX M
```

```
1 BIRT
2 DATE 7 APR 1949
1 FAMC F32
0 NOTE end of individual Mark Ardis
```

The first level 0 record identifies a new individual with id "I27". All following records with level greater than 0, until the next level 0 record, identify characteristics of individual I27. The "1 NAME" record specifies that the name of individual I27 is 'Mark Ardis'. (The individual's last name is surrounded by slashes.) The '1 SEX M' record specifies that the gender of individual I27 is male. The '1 BIRT' record is the first of two records that specify individual I27's birth date. The '2 DATE 7 APR 1949' record specifies the date and since level 2 is greater than level 1, the DATE record is associated with the preceding BIRT record. Finally, the '1 FAMC F32' record tells us that the preceding DATE and BIRT records are complete and that the current individual is a child in family F32 (Note that F32 may or may not be defined when this record is parsed.) Finally, the '0 NOTE' record tells us that the definition of individual I27 is complete (because we found another level 0 record).
Note that IDs of both individuals and families are arbitrary strings. You should ***NOT*** assume that all IDs have any specific format. I will test your program with you a variety of id formats.

The Wikipedia entry for GEDCOM <http://en.wikipedia.org/wiki/GEDCOM> has a nice introduction with pointers to more information. Family Echo <http://www.familyecho.com> and Geni <http://www.geni.com> allow you to create your family tree online and export it later in GEDCOM format. They are free, but they require you to create an account. Geni has a social networking component to its product. It will email you with reminders of birthdays and anniversaries of your relatives. You may want to try out one of these systems to see how information is recorded in GEDCOM files.

For this project, we are going to work with a subset of GEDCOM, and we will assume that all records are syntactically well-formed. That is, all records will start with a level number in the first character of the record, have a legal tag, and will have arguments in the proper format. Also, only one blank space will be used to separate all fields.

Here is a table describing all of the tags needed for the project:

Level	Tag	Arguments	Belongs to	Meaning
0	INDI	Individual_ID	top level	Define a new Individual with ID Individual_ID
1	NAME	String with surname delimited by "/"s	INDI	Name of individual
1	SEX	"M" or "F" (without the quotes)	INDI	Sex of individual
1	BIRT	none	INDI	Birth of individual. Typically followed by 2 DATE record that specifies the date.
1	DEAT	none	INDI	Death of individual. Typically followed by 2 DATE record that specifies the date.
1	FAMC	Family_ID	INDI	Individual is a child in family with Family_ID
1	FAMS	Family_ID	INDI	Individual is a spouse in family with Family_ID
0	FAM	Family_ID	top level	Define a new family with ID Family_ID
1	MARR	none	FAM	Marriage event for family. Typically followed by 2 DATE record that specifies the date.
1	HUSB	Individual_ID	FAM	Individual_ID of Husband in family
1	WIFE	Individual_ID	FAM	Individual_ID of

				Wife in family
1	CHIL	Individual_ID	FAM	Individual_ID of Child in family
1	DIV	none	FAM	Divorce event for family. Typically followed by 2 DATE record that specifies the date.
2	DATE	day, month and year in Exact Format	BIRT, DEAT, DIV, or MARR	Date that an event occurred
0	HEAD	none	top level	Optional header record at beginning of file
0	TRLR	none	top level	Optional trailer record at end of file
0	NOTE	any string	top level	Optional comments, e.g. describe tests

Exact Format for dates is a triple: *<day> <month> <year>*, where:

- the fields are separated by a single space
- *<day>* is the day of the month (with no leading zeroes)
- *<month>* is a 3-character abbreviation for the month (`JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC`)
- *<year>* is the year in 4 digits (e.g., 2010)

Here's an example of the GEDCOM data for a single generation of an imaginary family:

```
0 NOTE one generation of an imaginary family
0 I01 INDI
1 NAME Joe /Smith/
```

Define Individual I01

```
1 BIRT
2 DATE 15 JUL 1960
1 SEX M
1 FAMS F23
1 DEAT
2 DATE 31 DEC 2013
0 NOTE define Jennifer Smith
0 I07 INDI
1 NAME Jennifer /Smith/
1 BIRT
2 DATE 23 SEP 1960
1 SEX F
1 FAMS F23
0 I19 INDI
1 NAME Dick /Smith/
1 BIRT
2 DATE 13 FEB 1981
1 SEX M
1 FAMC F23
0 I26 INDI
1 NAME Jane /Smith/
1 BIRT
2 DATE 2 JUN 1983
1 SEX F
1 FAMC F23
0 F23 FAM
1 MARR
2 DATE 14 FEB 1980
1 HUSB I01
1 WIFE I07
1 CHIL I19
1 CHIL I26
```

The DATE record is associated with the preceding BIRT record

Family F23 has a husband, wife, and two children

Family F23 ends with the last record in the file

A summary of the family might look like:

Individuals

ID	Name	Gender	Birthday	Age	Alive	Death	Child	Spouse
I01	Joe /Smith/	M	1960-07-15	53	False	2013-12-31	None	{'F23'}
I07	Jennifer /Smith/	F	1960-09-23	56	True	NA	None	{'F23'}
I19	Dick /Smith/	M	1981-02-13	35	True	NA	F23	NA
I26	Jane /Smith/	F	1983-06-02	33	True	NA	F23	NA

Families

ID	Married	Divorced	Husband ID	Husband Name	Wife ID	Wife Name	Children
F23	1980-02-14	NA	I01	Joe /Smith/	I07	Jennifer /Smith/	{'I19', 'I26'}

We will assume that:

- The four records that require a date (`BIRT, DEAT, DIV, MARR`) will always be followed by a `DATE` record.
- The sex and birth date of every individual will be specified exactly once. (You cannot change your sex.)
- For each family specified in the file, the marriage and all family members will be specified.
- Each individual will be linked to every family where they are a child, for all those families that are described by the GEDCOM file.
- Each individual will be linked to every family where they are a spouse, for all those families that are described by the GEDCOM file.

We will make some assumptions when records are missing from the file:

- We will assume that an individual is alive if there is no `DEAT` record.
- We will assume no divorce has occurred for a marriage if there is no `DIV` record.
- Some families may not be specified, since the parents of the family are not specified in the file.
- This last condition describes the top of the genealogical tree---it has to stop somewhere.

The Mormons have a conservative view of families. Every family has one husband, a male, and one wife, a female. A marriage ends either by divorce or the death of at least one

spouse. If a widow or widower remarries then she/he becomes a member of a new family (as the wife/husband).

The program you will write will look for errors and anomalies in a GEDCOM file and display a simple summary of the data in the file. ***Errors*** are combinations of records that cannot logically all be true. Here are some examples of errors:

- Death date occurring before birth date
- Marriage of a dead person
- Female husband in a family
- Note that you get no credit for detecting syntax errors. All data is assumed to be syntactically correct.

Anomalies are combinations of records that appear to be erroneous, but might actually be true. Here are some examples:

- Birth of a child before his/her parents are married (potentially embarrassing if true)
- Being a spouse in two marriages at the same time (polygamy, illegal in most places)

It is up to your team to choose which errors and anomalies you will detect and report from a master list of user stories found in the ***TeamXXReport.xlsx*** file. You do not need to repair any of the errors or anomalies in the GEDCOM file, just point them out. When you report errors and anomalies you must include the unique ID and name of all individuals involved, and the unique ID of all families involved in these errors/anomalies. For example,

```
Error US03: Birth date of Harpo Marx (I03) occurs after his
death date.

Anomaly US08: Birth date of Chico Marx (I02) occurs before
the marriage date of his parents in Family F05.
```

It is also up to your team to decide how you will display a summary of the data in the GEDCOM file. As a minimum, you should include a list of all the individuals found (with their basic information, such as unique ID, name, sex and birth date) and a list of all the families, showing names of family members. The master list of user stories includes additional items to list. Note that you are not asked to produce graphical versions of family trees, though you could produce an indented-list style if you wish.

The project will be delivered in four sprints where each sprint adds functionality to

previous sprints.

Your program must run as a command from a terminal window that expects one GEDCOM file as input. Note that this file will be specified on the command line---you cannot hard code its name. All output should be printed in ordinary ASCII characters to the terminal window. You will have to save this output in a single text file for assignment submissions.

Your solution should be tested against an input file that your team creates and GEDCOM files that I will provide for your team.