Lab 10 Option 1
Linked List and Binary Search Tree
due: April 29, 2016 11:59pm

Important Notes

- Submission repository: /rd/cs246s2016/$USER/lab10
- This is a hard deadline. No late submission is allowed.
- **This assignment can be done in pairs or groups.** If you need help, see the instructor or TA.
- Read through this specification completely before you start.
- Some aspects of this specification are subject to change, in response to issues detected by students or the course staff.

1 Introduction

This Lab is based on your projects 6 and lab 9 option 1 using the baby name data files. Your program is going to print out the horizontal histogram describing the popularity of a particular name. For example, the following histogram shows that over all the years, 15703 female babies were named "Eden" with the total rank 373. Before 1990, "Eden" was not a popular name at all (at least not top 1000). After that, the popularity of "Eden" has been increasing and reached its peak time in 2012 where 1909 babies were given this name and it was ranked 164 over all female names.

```
Eden : total rank (373), total count (15703)
Histogram------------------
1990  133(724)
2000              505(524)
2001                  722(397)
2002                      797(373)
2003                          899(343)
2004                          922(341)
2005                          987(321)
2006                             1023(317)
2007                                     1342(257)
2008                                         1479(229)
2009                                             1560(206)
2010                                                 1701(180)
2011                                                 1724(181)
2012                                                         1909(164)
```

In this project, for each gender and each distinct name, you are going to represent it with the following two types of structures:

```
struct totalRank {
    //Structure to hold the total rank for each individual
    //There will be one instance per individual
    //It links to the year rank
    int rank; //total rank
    char name[NAME_LEN+1];
    int totalcount; //total count
    struct yearRank *yearPointer;
};
typedef struct totalRank DNode;

struct yearRank {
    //Structure to hold the year rank for each individual
    int year;
    int rank;
    int count;
    struct yearRank *next;
};
typedef struct yearRank YearNode;
```

For every gender and every name, the rank information for each year (if the name appears in the corresponding file) and for the total rank can only be stored once. As an example, the information for female name "Eden" will be saved like this:
`[373, "Eden", 15703]`→`[1990, 724, 133]`→`[2000, 524, 505]` →`[2001, 397, 722]`→
`[2002, 373, 797]`→`[2003, 343, 899]`→`[2004, 341, 922]`→ `[2005, 321, 987]`→
`[2006, 317, 1023]`→`[2007, 257, 1342]`→`[2008, 229, 1479]` →`[2009, 206, 1560]`→
`[2010, 180, 1701]`→`[2011, 181, 1724]`→`[2012, 164, 1909]` →`NULL` where the first node is of type `DNode` and the rest are all of type `YearNode`. Note that you do not store any dump node where the given name does not appear in that year.
Your goal is to use linked lists and binary search trees to organize these lists by gender to facilitates fast searching, in other words, to build indices on the data for searching. There are two ways of searching a name: by the total rank or by name. This means that you need to build two indices for all lists for each gender.

2 Tasks

1. Process 25 name files and populate data into your data structures. For each name and each gender, the data structure is already given. You need to find a way to organize all names for each gender. At the same time, build indices for the data, one has name as the search key and the other is the total rank. Your program should not waste any space by storing the same information in different data structures. Your binary search tree should be built balanced so that the complexity of searching is $\Theta(logN)$ where N is the number of distinct (male/female) names.

   After you are done with this part, test it right away. If two names (with same gender) have the same total count, they will have the same rank. For example, the total ranks of male names whose total counts are 28, 27 or 26 are listed below for your information.

   ```
   2380 Primitivo 28
   ```

```
2380 Willy 28
2380 Windell 28
2380 Sigurd 28
2380 Virginia 28
2380 Epifanio 28
2380 Price 28
2380 Adelard 28
2380 Friedrich 28
2380 Dorris 28
2390 Lew 27
2390 Candelario 27
2390 Vidal 27
2390 Toy 27
2390 Arlis 27
2390 Pink 27
2390 Bishop 27
2390 Davie 27
2390 Carolyn 27
2390 Rose 27
2390 Mervyn 27
2401 Ephraim 26
2401 Fortunato 26
2401 Okey 26
2401 Alonza 26
2401 Wellington 26
2401 Heber 26
2401 Hurley 26
2401 Arno 26
2401 Lillian 26
2401 Emma 26
2401 Amador 26
2401 Mearl 26
2401 Brown 26
2401 Buddie 26
2401 Sid 26
2401 Dionisio 26
```

2. To search and print out the histograms, you have the following options (flags) which will be implemented by using command line arguments.

   - `-g gender` : search for the female data if "gender" is F or f and male data if "gender" is M or m.
   - `-r rank` : search by rank where "rank" is a positive integer that stands for the total rank. All names with the same rank have to be printed out, one histogram per name.
   - `-n name` : search by name where "name" is a string that represents a name.

   If there is no `-g` option, then information has to be printed for both genders.

For example, if a user runs the program with options "-g f -n Eden -r 20", your program should output (order does not matter):

```
Alyssa : total rank (20), total count (138468)
Histogram------------------
1960  68(923)
1970  296(395)
1980     1648(89)
1990                    4929(23)
2000                                                                  13444(12)
2001                                                               13148(14)
2002                                                               12747(12)
2003                                                               12688(14)
2004                                                            11975(14)
2005                                                        10726(16)
2006                                                    10134(19)
2007                                                            11146(14)
2008                                                     9607(16)
2009                                          7935(19)
2010                                  6934(20)
2011                              5989(37)
2012                    5054(44)

Eden : total rank (373), total count (15703)
Histogram------------------
1990  133(724)
2000              505(524)
2001                  722(397)
2002                      797(373)
2003                          899(343)
2004                          922(341)
2005                          987(321)
2006                             1023(317)
2007                                     1342(257)
2008                                         1479(229)
2009                                             1560(206)
2010                                                 1701(180)
2011                                                 1724(181)
2012                                                         1909(164)
```

If a user runs the program with options "-r 2380", your program should output

```
Check for the male baby data.
Primitivo : total rank (2380), total count (28)
Histogram------------------
1930  28(953)

Willy : total rank (2380), total count (28)
```

```
Histogram------------------
1900  28(945)

Windell : total rank (2380), total count (28)
Histogram------------------
1940  28(996)

Sigurd : total rank (2380), total count (28)
Histogram------------------
1900  28(942)

Virginia : total rank (2380), total count (28)
Histogram------------------
1930  28(958)

Epifanio : total rank (2380), total count (28)
Histogram------------------
1930  28(947)

Price : total rank (2380), total count (28)
Histogram------------------
1900  28(939)

Adelard : total rank (2380), total count (28)
Histogram------------------
1900  28(924)

Friedrich : total rank (2380), total count (28)
Histogram------------------
1900  28(932)

Dorris : total rank (2380), total count (28)
Histogram------------------
1930  28(944)

Check for the female baby data.
There is no baby name with this rank.
```

The scale you use to plot the histogram is entirely upto you, the only requirement is that it fits the screen. Typical unix terminal window is 80x24, you may use larger sizes as most windows are resizable, but make sure you do not exceed reasonable screen sizes. 80 characters is usually about half of the screen size on most monitors these days, so 160 characters really should be about as large as the width should get.

3. Error handling. With any invalid input, your program should prompt usage for the users. For example, with invalid options "-r -n Eden", a message might be:

```
Invalid option.
```

```
Usage: lab10 [-g gender] [-r rank] [-n name]
  g: search based on gender. F or f stands for female; M or m stands for male.
  r: search by rank.
  n: search by name.
```

4. Clean up the memory. You need to free all the dynamically allocated space.

3 What to submit

- Your source code, including both .cpp and .h files. Make sure your files are well organized.
- A readme file. In your readme file, you need to explain
  1. what data structure you use for building index on total ranks,
  2. what data structure you use for building index on names,
  3. what is the complexity of searching by name for your program,
  4. what is the complexity of searching by rank for your program,
  5. what algorithm(s) you use for freeing memories,
  6. how to run your program,
  7. what a user should expect from your program.

  If your program does not fulfill some requirement(s), you need to make a remark in this document.
- A makefile.