Zach Sawicki

Project 2

User Guide

To run this program, simply type (PROGRAM NAME), followed by a space and either “-r" followed by a space and the name of the file to read in a field delimited file and convert it to length indicated, or “-z” followed by the three to five digit zip to search for that specific zip. If the zip does not exist, the program will notify you of this.

Examples:

(PROGRAM NAME).exe -r file.csv

file imported successfully

Do you want to search the database? (Y/N): y

enter a zip: 56303

Zip Code: 56303 Place Name: Saint Cloud State: MN County: Benton Lat: Long:

(PROGRAM NAME).exe - z 56301

Zip Code: 56301 Place Name: Saint Cloud State: MN County: Benton Lat: Long:

Design Document

**Introduction**

This program receives a data file with every zip code, place name, county, state code, latitude and longitude in the United States. The program calculates the most east, most west, most south and most north zip codes of each state. Once that is complete the data is length indicated and stored in a new file. In addition to this, the zip code and byte offset of each record is calculated and stored in an index file. The user then can search for a zip code and the record will be retrieved from the length-indicated data file.

**Classes**

This program has four classes. Those are zip, delimBuffer, LIBuffer, and primaryindex. These are all used to

The first mentioned class is zip, and is a remnant from the first iteration of this program. It simply encapsulates the data required for each zip into an instance of the class.

Second, the delimBuffer is used to correctly read the data from the csv and properly sort it to be placed onto each data member of the zip classes.

Next, LIBuffer takes the data that was read in by the delimBuffer, and adds the length of each record to the beginning of each record. It then writes this to a new file.

The final class that was mentioned was the primaryindex class. With the help of a structure called indexElement, the class successfully keeps the indices sorted by the primary key, zip code.

**Data Structures**

The first data structure is a vector of vector zips called states. This data structure is called in the readCSV function and is used to contain all of the data from the data file. The first vector has 57 elements and because there are 57 different zip codes. Then the secondary vector contains each of the zip objects for each record.

The second data structure is called indexElement and it is a struct which contains the zip code and their byte offset in the length-indicated data file.

The third data structure is a vector of the indexElement structs called index.It houses all of the zip codes and their byte offset within the length-indicated file. This is allocated in the primaryIndex class. The vector is sent to the LIBuffer class to be put into the index file.

The fourth data structure is a vector of strings called headerElement. This is used to store the header data given from the original data file. The data from the file is delimited with commas so each field is stored in its own position in the vector. The vector is then used to match the field with the incoming record data.

The fifth data structure is a pointer to a character array called argv. This structure is used for obtaining the command line arguments. The first thing it should store is the name of the in file and also the zip code that the user wants the associated record for.

**Functions**

The zip class contains 16 functions. The first function is a default constructor, the second function is a specified constructor where the zip code, city, state code, county, latitude and longitude are specified. The third function is a copy constructor. There are six functions that set each data value in the zip class. One for zip code, one for city, one for state code, one for county, one for latitude and one for longitude. There are six more functions that retrieve the value of all six of private members. The final function is called print and it displays the contents of a class instance to the console.

The LIBuffer class contains 8 functions. The first function is a default constructor the second function is a specified constructor which sets the delimiter to a comma and the max size of the buffer. The third function is called read and it reads from the length-indicated data file with a given offset and returns the associated data record. The fourth function is called write and it writes data to the length-indicated file. The fifth function is called unpack and it unpacks the buffer and returns a single field from the buffer. The sixth function is called pack and it adds a record field to the buffer. The seventh function is called getBuffer and it returns the string buf from the class. The final function is called getSize and it returns the size of the buf string.

The delimBuffer class contains 6 functions. The first function is a default constructor the second function is a specified constructor which sets the delimiter to a comma and the max size of the buffer. The third function is called read and it reads from a given data file and puts a line from the file onto the buf string. The fourth function is called unpack and it returns a single field from the buf string. The fifth function is called setBuffer and it sets the buf equal to a given string. The sixth function is called getBuffer and it returns the string buf.

The primaryindex class contains 18 functions. The first is a default constructor, followed by two specified constructors. An add function adds an index and keeps the indices sorted by zip code. A binSearch function takes in an integer zip as a parameter and returns the offset as an unsigned long integer. It uses a helper function called search that calls the binSearch function and passes the target zip to it. A writeToFile writes the indices to the harddrive. A readIndex reads the written indices back into memory. A readCSV creates a vector of vectors of zip objects, calls another function called transfer, calls writeToFile, and opens up the two data member stream objects. printTable is a remnant from the first project, and prints a table of the most north, south, east, and west zips of each state/region. It works with the help of five functions: stateChooser, mostNorth, mostSouth, mostEast, and mostWest. The stateChooser works by placing the correct state code given by the vector of vector index. The most functions work by comparing either latitude values for east and west, and longitude values for north and south. readIn correctly parses through the original csv, making sure to remember the order of fields, and builds the length indicated csv with this information.

**Main Program**

The main program has three different avenues based off of the command line arguments. The first and simplest option is that the user gave an invalid command. The second option is that the user input something to the effect of “-r filename.ext”. The program then opens the given file as well as an index file and a length-indicated data file. An instance of the primaryIndex class is declared which is given the user’s file. Declaring the primaryIndex class instance in this way automatically calls the readCSV function in its constructor. After this is completed the user is given the option of searching the length-indicated file. If the user selects yes they are prompted to give a zip code. Once the zip code is input the index file is searched for that zip code and its associated byte offset. The byte offset is then used to find the correct position in the length-indicated data file. The line is read from the position which contains the record for the user-input zip code. The record string is sent to the LIBuffer unpack function and each field's data is displayed back to the console.

The third option is that the user inputs “-z (zip code)”. This means the user wants to search the already built length-indicated file for the associated record. This is done by constructing an instance of the primaryIndex class with the given parameters of the index file and the length-indicated data file. Declaring the class instance this way automatically calls the readIndex function. The readIndex function builds the primaryIndex index vector with all of the zip codes and byte offsets. Then back in the main function the given zip code is passed to a search function to find its associated byte offset. The byte offset is then used to find the correct position in the length-indicated data file. The line is read from the position which contains the record for the user-input zip code. The record string is sent to the LIBuffer unpack function and each field's data is displayed back to the console.

Test Document

Test 1 -r Tag

D:\Users\Jacob\Desktop\CSCI 331\Assignments\Group Project 2\Source\CSCI\_331\_GP2\_T2\x64\Debug\CSCI\_331\_GP2\_T2.exe -r us\_postal\_codes\_ROWS\_RANDOMIZED.csv

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*State |East |West |North |South \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*AA |34034 |34034 |34034 |34034 \*

\*AK |99522 |99830 |99722 |99830 \*

\*AL |35060 |36275 |35979 |35060 \*

\*AP |96337 |96507 |96507 |96337 \*

\*AR |72670 |72543 |72543 |71962 \*

\*AZ |85085 |85721 |85085 |85721 \*

\*CA |93446 |92308 |93446 |92308 \*

\*CO |81621 |80817 |81621 |80817 \*

\*CT |6911 |6251 |6251 |6911 \*

\*DC |20016 |20061 |20016 |20061 \*

\*DE |19956 |19902 |19718 |19956 \*

\*FL |33709 |32136 |32136 |33709 \*

\*FM |96941 |96941 |96941 |96941 \*

\*GA |30162 |30453 |30453 |31744 \*

\*HI |96782 |96761 |96782 |96761 \*

\*IA |51638 |50611 |50611 |51638 \*

\*ID |83548 |83252 |83548 |83252 \*

\*IL |61244 |62058 |60087 |62058 \*

\*IN |46923 |46186 |46507 |46186 \*

\*KS |67529 |67013 |67529 |67013 \*

\*KY |42047 |40741 |42047 |42104 \*

\*LA |71260 |70377 |71260 |70377 \*

\*MA |1852 |2673 |1852 |2673 \*

\*MD |20763 |20627 |20763 |20627 \*

\*ME |4950 |4553 |4950 |4553 \*

\*MH |96970 |96960 |96970 |96960 \*

\*MI |49643 |48421 |49643 |48421 \*

\*MN |56056 |56452 |56452 |56056 \*

\*MO |64461 |65050 |64461 |65050 \*

\*MP |96952 |96950 |96951 |96952 \*

\*MS |38950 |38863 |38950 |39337 \*

\*MT |59724 |59211 |59252 |59211 \*

\*NC |27570 |28509 |27570 |28509 \*

\*ND |58249 |58237 |58249 |58237 \*

\*NE |68780 |68379 |68780 |68379 \*

\*NH |3241 |3263 |3561 |3263 \*

\*NJ |8057 |7934 |7934 |8057 \*

\*NM |87047 |88323 |87047 |88323 \*

\*NV |89430 |89142 |89430 |89142 \*

\*NY |12701 |12485 |12485 |12701 \*

\*OH |45230 |43701 |44055 |45230 \*

\*OK |73006 |74055 |74055 |73006 \*

\*OR |97347 |97228 |97228 |97347 \*

\*PA |15083 |17237 |15868 |17237 \*

\*PW |96940 |96940 |96940 |96940 \*

\*RI |2829 |2818 |2876 |2818 \*

\*SC |29743 |29532 |29743 |29002 \*

\*SD |57636 |57217 |57217 |57636 \*

\*TN |38052 |38552 |38552 |38052 \*

\*TX |78293 |77209 |79172 |78293 \*

\*UT |84330 |84535 |84330 |84535 \*

\*VA |23226 |23432 |23226 |23432 \*

\*VT |5459 |5602 |5459 |5602 \*

\*WA |98644 |98356 |98257 |98356 \*

\*WI |54645 |53149 |54914 |53149 \*

\*WV |25705 |26217 |25705 |25920 \*

\*WY |82943 |82643 |82431 |82643 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*State |East |West |North |South \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

file imported successfully

do you want to search the database? (Y/N):

Please enter a valid zip: Target hit!: 56301 912513

Zip Code: 56301 Place Name: Saint Cloud State: MN County: Stearns Lat: 45.541000 Long: -94.181900

Test 2 -z Tag

D:\Users\Jacob\Desktop\CSCI 331\Assignments\Group Project 2\Source\CSCI\_331\_GP2\_T2\x64\Debug\CSCI\_331\_GP2\_T2.exe -z 56304 40934 records in the file.

Target hit!: 56304 927284

Zip Code: 56304 Place Name: Saint Cloud State: MN County: Stearns Lat: 45.552101 Long: -94.128403

Hello World!