Design Document

Title: Address Book Program, Brian Pina, 104790387, 03-25-2016

Problem Description: This program will manage a Address Book database, which can organize, update, sort, search through, and print as needed. Each part of the database holds information on one subject, the information held is unique id, first name, last name, company name, home phone, office, phone, email, mobile phone, address, city, state, zip code, country, list of affiliates. The base structure will be an AVl tree while all other data will either be held in a array or vector.

Architecture: My program will initially use mainMenu function that will present the user with a menu that will do commands of their choosing, update, organize, sort, search,print, this function will return a value that the user chose and will get passed to a menu choice function which will call each corresponding function, print will call prinBook function and will print the address book each function will have their own sub menu which will do their own actions and will call their own corresponding functions.

Print Book will call print menu

SourceBook will call sort menu

Organize will call Organize menu

Search Book will have a initial menu which will do a first search and then the user will have an option to do a second search or start their search over,

Update book will call update menu where the user will choose to add delete or update a field

To update a update or delete field the search function will be used here and for adding the program will look for a nullptr or empty field.

Input Requirements:

This program will receive input for each menu introduced such as main menu and all the submenus for each function.

The user will input numbers from 1 - 2 for yes or no and also numbers for what fields they would like to manage or see, other input would be certain data fields such as strings or ints that they will use in the search function. There will be error proofing if user enters the wrong input they will be asked to enter right info or exit.

If a string is entered instead of a int in a sort after specifying the data type they will be asked to enter input again. The program will read in a predefined address list of 50 contacts.

Phone numbers will not surpass 10 numbers emails will be delimited by a logical length and names will only hold 15 characters each.

Output:

The program will output menus that the user will use as a reference to enter data, each menu will call functions that have their own submenus, Data output will be each entry in the tree that will get added to the file, the output will get written to a file and each field and subject has its own delimiters that will separate each field or subject. We will be writing to a ofstream file that will have pre constitutions were we have the obvious if it doesn't open the program stops, other that we will proof check by making sure the right delimiters are added and that the fields entered are of the right value and length.

Problem Solutions:In my algorithm i've included very odd ways of problem solving where i created a denomination function for my print function, it will hold a array that will receive numbers that will help in showing what the user will want to print out, let's say email and phone number but not the company name, either that or implementing a recursive function that will ask for each field they would want to print out.

Data Structure:

I choose to use a queue to hold the data input from file so that the address books gets filled in before it passes to the tree from there we will copy over each iteration until the queue is empty by using its pop function, and from there the tree should be completely filled, in my affiliates member I chose a vector since there isn't a predefined amount of affiliates so a vector was chosen for its flexibility and ability to resize for when we want to delete a member, In my denominations function having a array that holds yes or no ints that the program will use is a weird form of algorithm that i got used to but it works and i'm very familiar with and it also will help with future recursion.

User Interface Scheme:

The menu is pretty self explanatory where each item will have a value starting from 1 in a ascending value, the submenus are laid out in the same way where error proofing is used just incase a wrong value is entered if a value if invalid we will, show the menu again and each menu will have an option to revert to the last menu or exit the program.