**User Manual**

**B+ Tree**

**S. Cloud State University**

**Team 12**

**Adcock Hunter, hm3101dq**

**Kushal Singh, vu4055bm**

**Sabin Basnet, bp4065pg**

**Ashish Dahal, xc0728qf**

**Mohammad Hejazi Najdi, xb8321ui**

**Working Process**

This is a B+ tree project that deals with manipulation of different records like zip code, name of a person, state name, name of cities, longitude and latitude. A B+ tree can represent an ordered list or unordered list, and in such an implementation, the operations of search, insertion, and removal are in general fast.

In this project, an interactive program will manipulate the records held in an unordered list, and the list will be encapsulated in a class and represented by a pointer-based B+ tree. As program being the interactive, it can read and respond the commands entered by the user. The program does all sorts of operations manipulating the list. For example:

Checks if the list is empty. Also

* Insert the item or records into the tree
* Remove the item or records from the tree or sequence set
* Reports the information through zip code
* Finally writes out the list in the terminal window

The program is all menu-driven. In the main program, we need to design in such a way that the user enters the character that decides which operation to run. Like if the user enters ***I,*** that means program should insert the entered record into the list and so on for remove and all.

This is how we designed to have our program to be but due to the lack of time and late updates, we were not able to add the menu and everything. But we did manage to read the file and print out the data, count the number of records and size of the sequence set.

**Location**

As we didn’t have access to the lab’s computers, so, we did in our own computers. In order to access for all the team members and others, here is the link to the GitHub repository as it is most professional and easiest way to work in group.

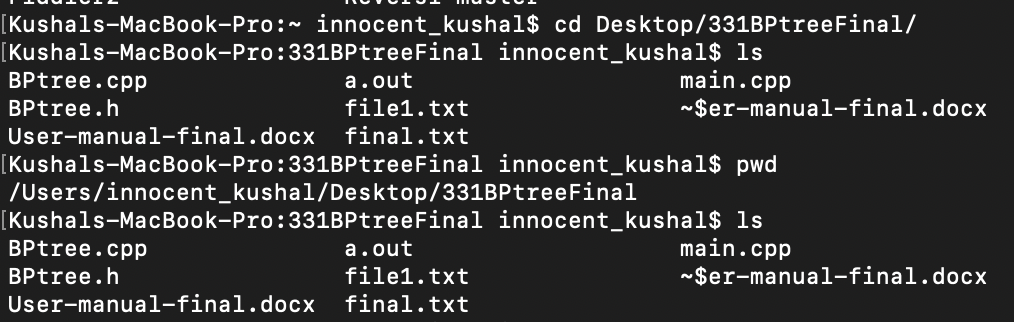
<https://github.com/XKushal/B-tree>

And in this computer, it is located in:

: **pwd**



**Detail:**



This process would be same for everyone in our group. So, we just included only one sample.

**Compile and Run**

After we are done with all the coding, we run the program and for that we need to compile the program. In the terminal window, first we got to find the files and then compile. Like in this case,first we go to working directory, which is made sure using the command pwd.

Kushals-MacBook-Pro:331BPtreeFinal innocent\_kushal$ pwd

/Users/innocent\_kushal/Desktop/331BPtreeFinal

After we are in the correct working directory, we need to compile our file. For that, we use the linux command g++ followed by filename. That is, in this case it would be:

Kushals-MacBook-Pro:331BPtreeFinal innocent\_kushal$ g++ main.cpp

As the name of our main file is main.cpp

If the program has any errors, it would show up in the terminal screen, otherwise the cursor will move to the next line, which means program is good to run. After we compile the program, we use another command **a**.**out** to run.

Kushals-MacBook-Pro:331BPtreeFinal innocent\_kushal$ ./a.out

Then as per the module of the program, user will be able to follow the program and do whatever the user wants to do.

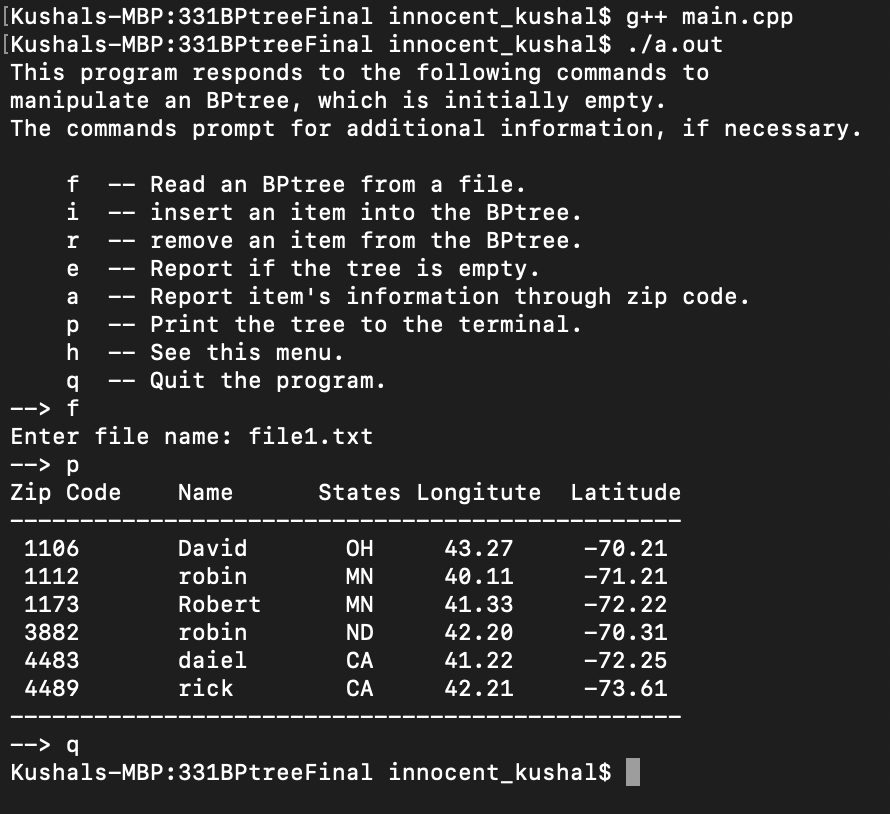
**Sample run**

This design is for the future implementation of our program. We were not able to formulate this version in our code, but this is what we thought our program would look like if we were able to complete everything.

**Sample 1:**

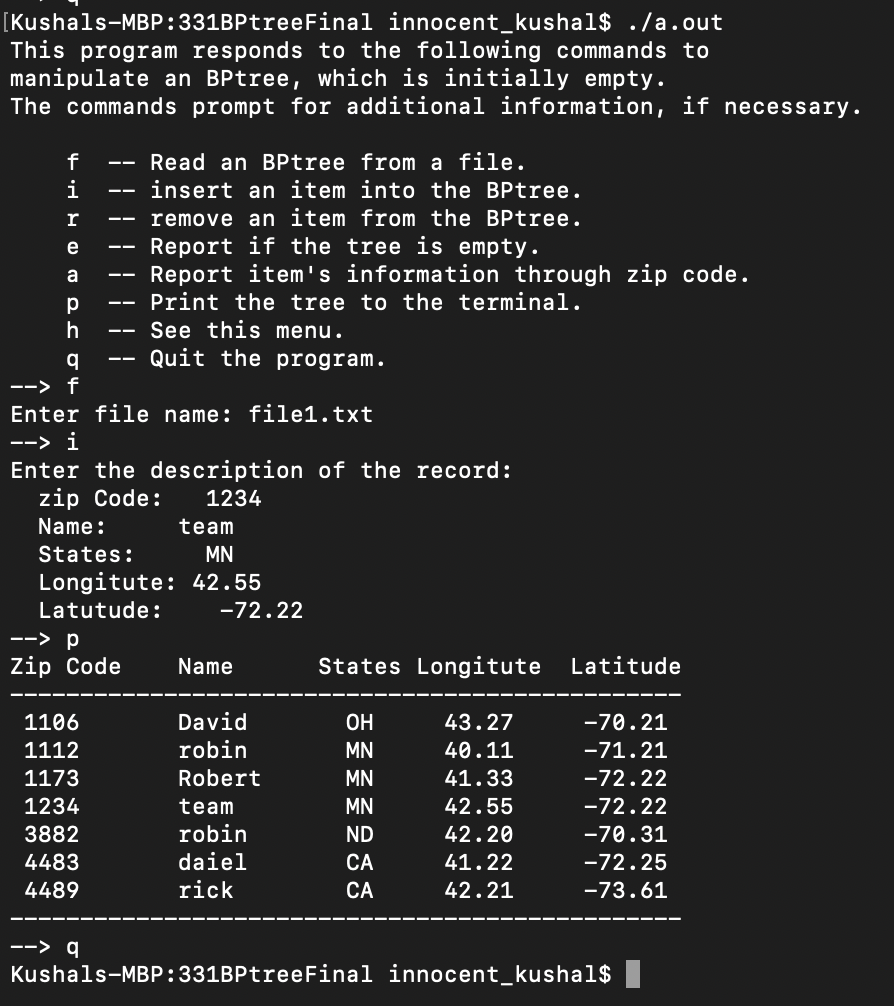
**After we run the program, this is how it looks.**

**It will show the menu and user will do as per their will.**

****

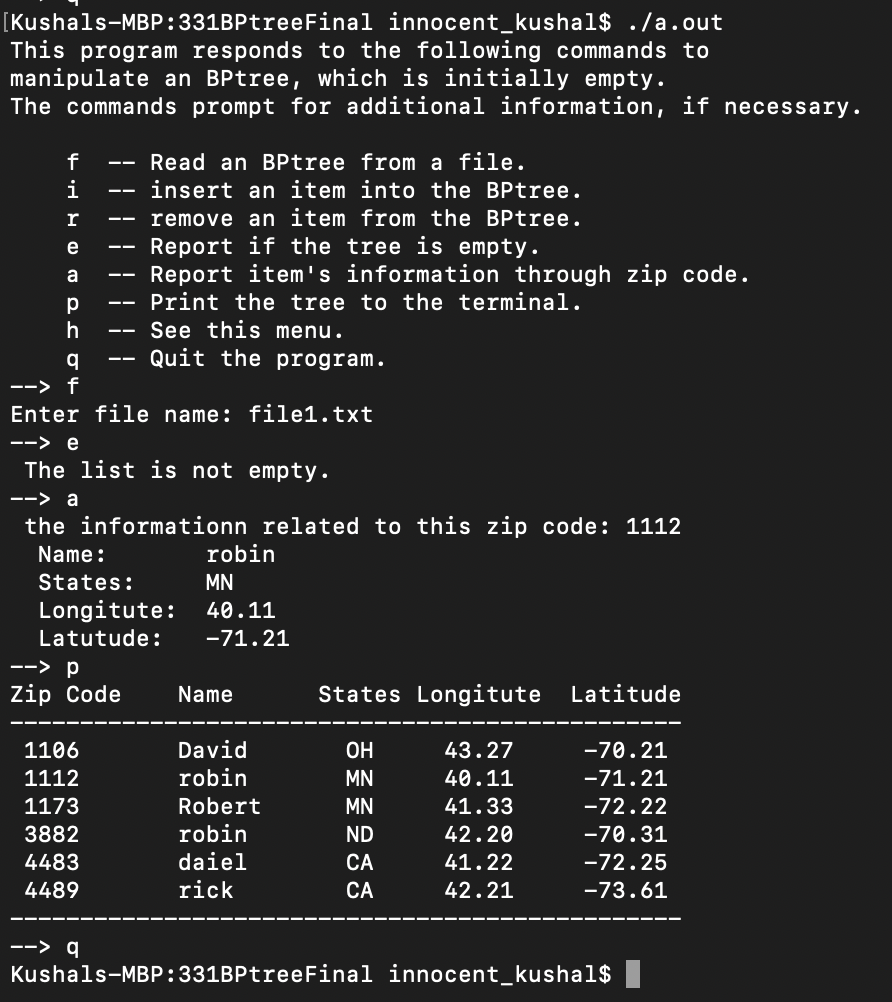
**Sample 2:**

**This sample shows that user can insert the value with a command I and quits the program with the command q. we can see after we insert the new value, its updated in the record.**

****

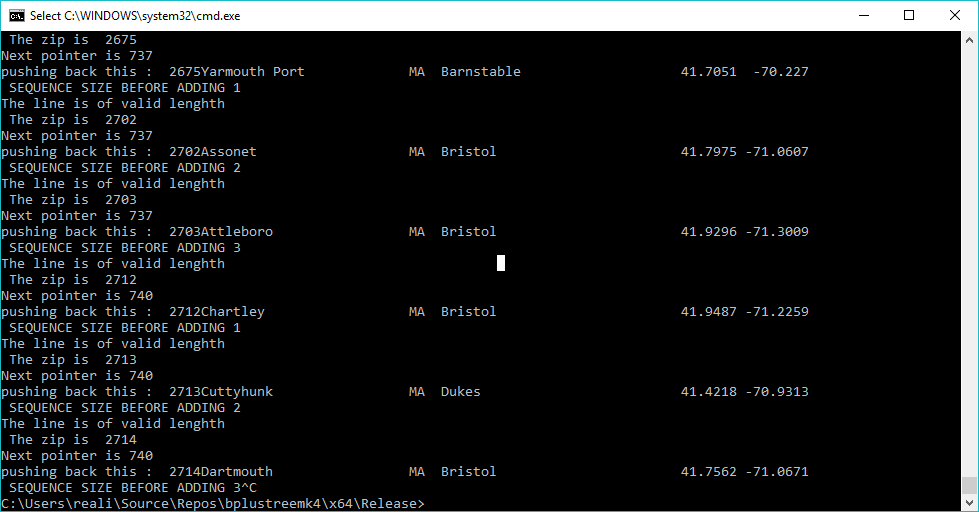
**Sample 3:**

**This sample checks if the list is empty with the command “e” and gives the detail of the record through the zip code.**

****

**Sample 4:**

**However, due to some circumstances, our run would look like this, where there are records that are being inserted to the sequence sets.**



**Known errors**

Before the run, we stuck into lot of errors like accessing the file, even reading the both integers and string both at the same time.

Also, another bug we ran into that was really crippling was when piping in data from the file the typical “myfile >> var1 >>var2 >>var3” failed after the second one even with proper delimiters and everything. So, to get data from a file and assign it to variables we had to go to really annoying lengths to circumvent this.