Using grep

By Jayviar Laporte

For Starters: We will be using our CentOS Server to do this Lab. Our File name is called GrepLab but your file name may look different depending on if you changed it after downloading it. In this Lab, we will be using and explaining how the grep command works. \*Make sure you are in the same directory as the file name you are trying to grep with.\*

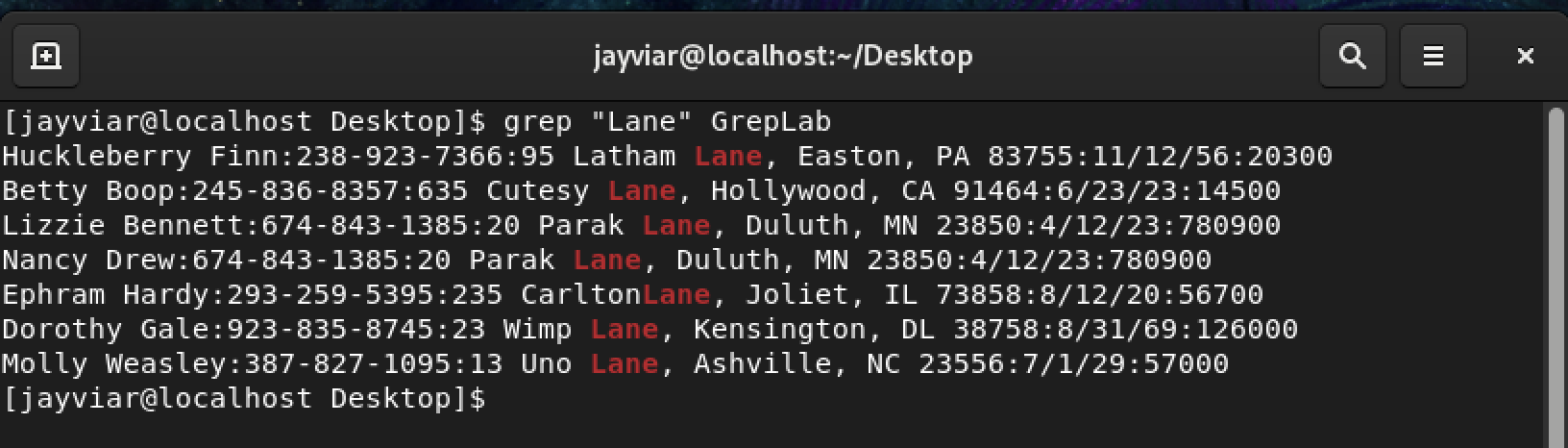
Grep Syntax:

grep “pattern” filename

1. Print all lines containing the string Lane

grep Lane GrepLab

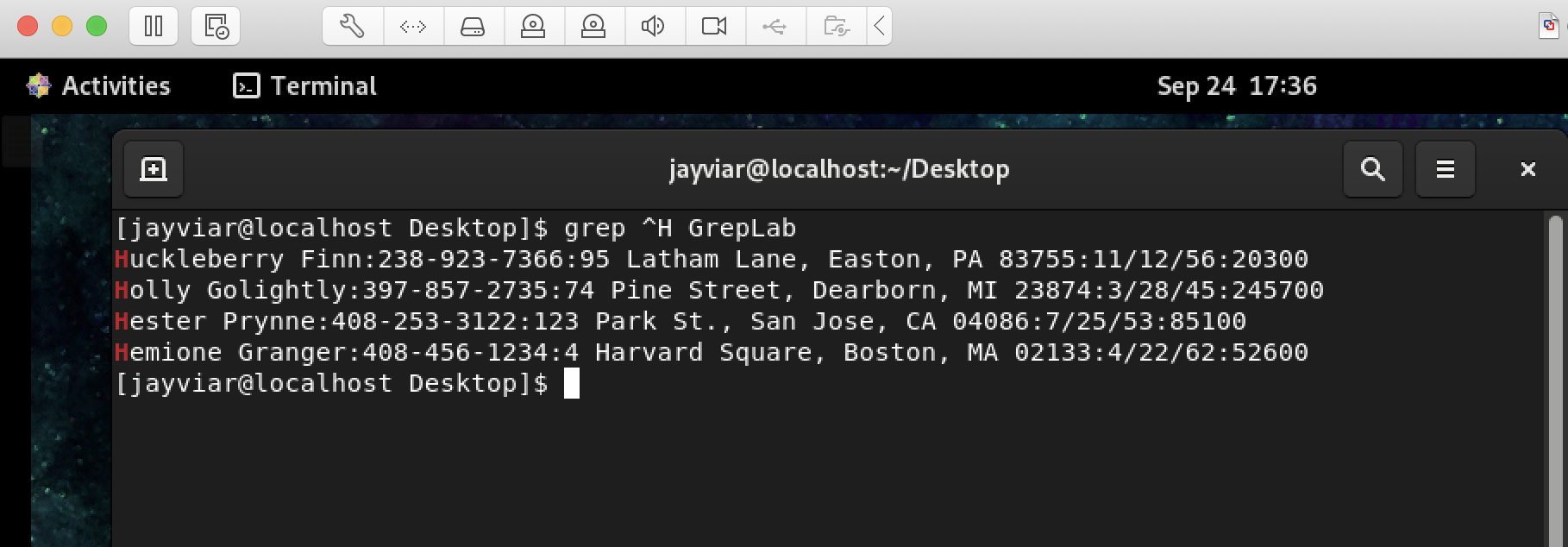
This works as grep will take every instance, and only the instance where this word occurs and return it back to us.



1. Print all lines where the person’s first name starts with

grep ^H GrepLab

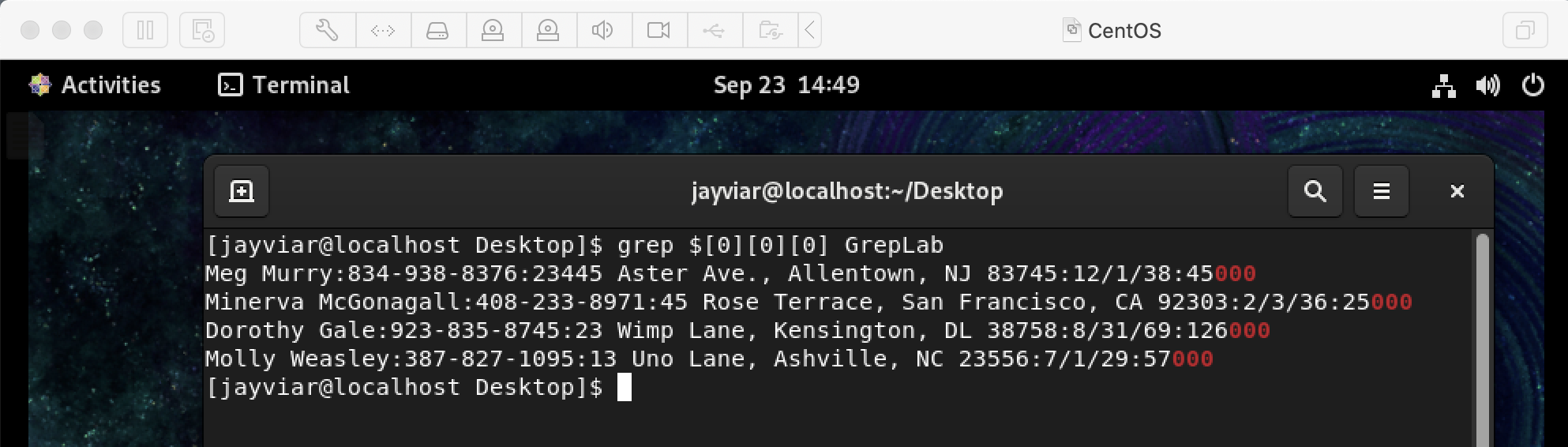
This code prints our solution because we know every line starts with the first letter of a person's first name. So, we use ^ to read lines that start with a capital H.



1. Print all lines ending in three zeros (000)

grep $[0][0][0] GrepLab

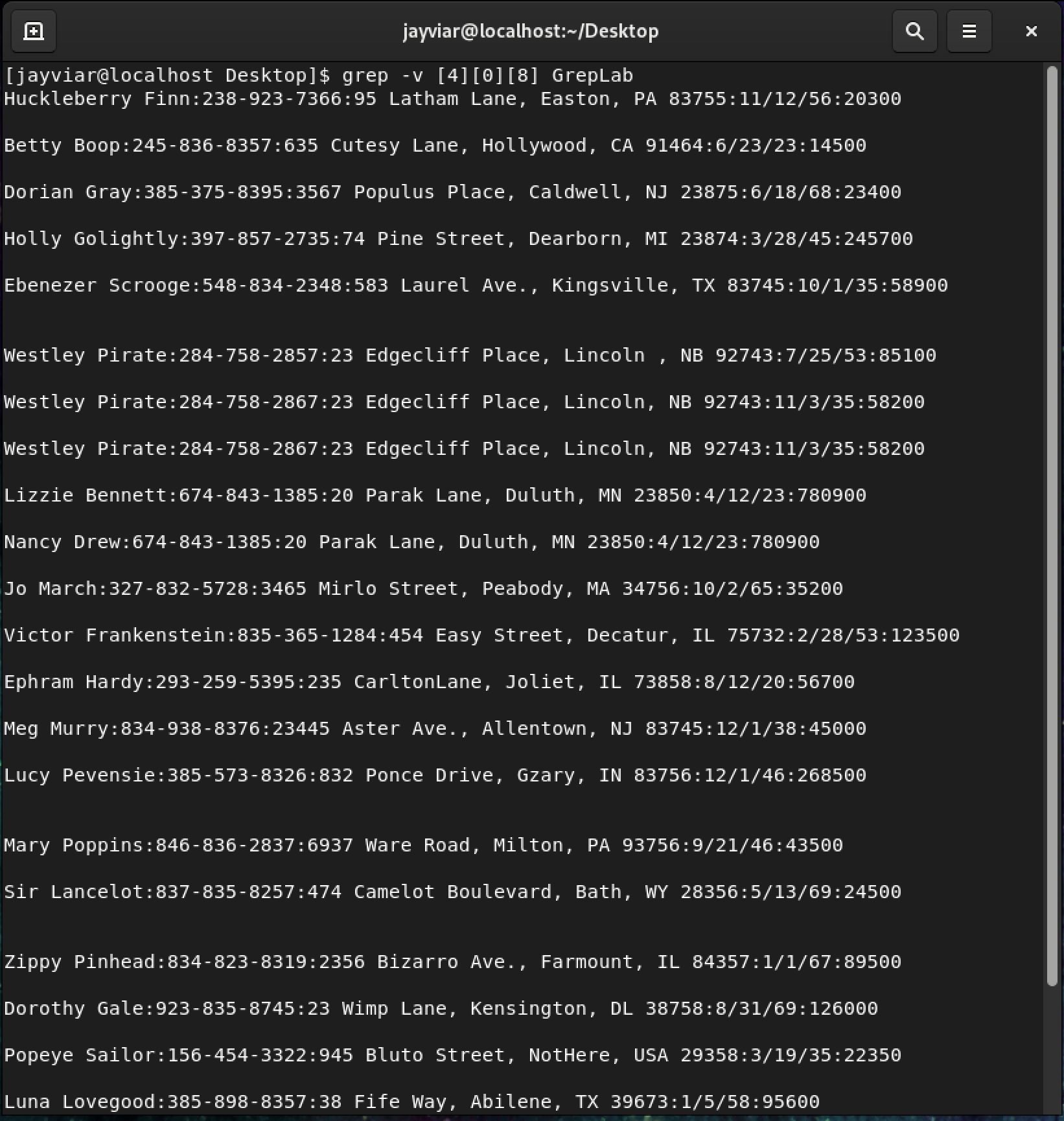
For this solution, we are using $ as it will match a pattern at the end of a line and then we use the brackets to specify the individual number we want to find rather than a range of them. We then put them together consecutively and get our answer



1. Print all lines that don’t contain 408

grep –v [4][0][8] GrepLab

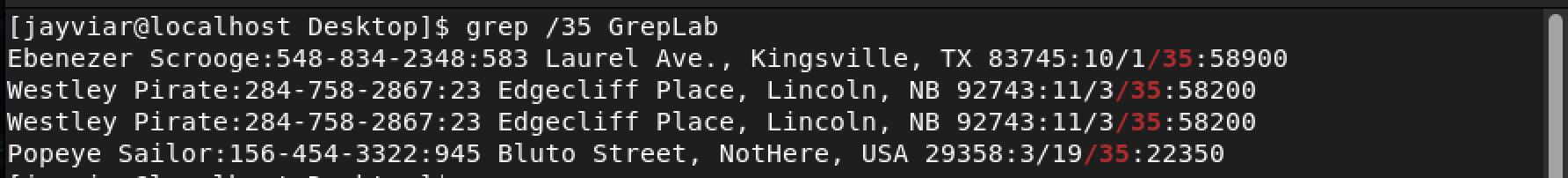
For this solution. We used the –v command in grep. Think of it as a sort of inverse command of whatever it is you want to find the NOT of. After we type –v, we go ahead and type our number 408 since we want everything that does \*not\* contain this sequence of numbers.



1. Print all lines where birthdays are in the year 1935

grep /35 GrepLab

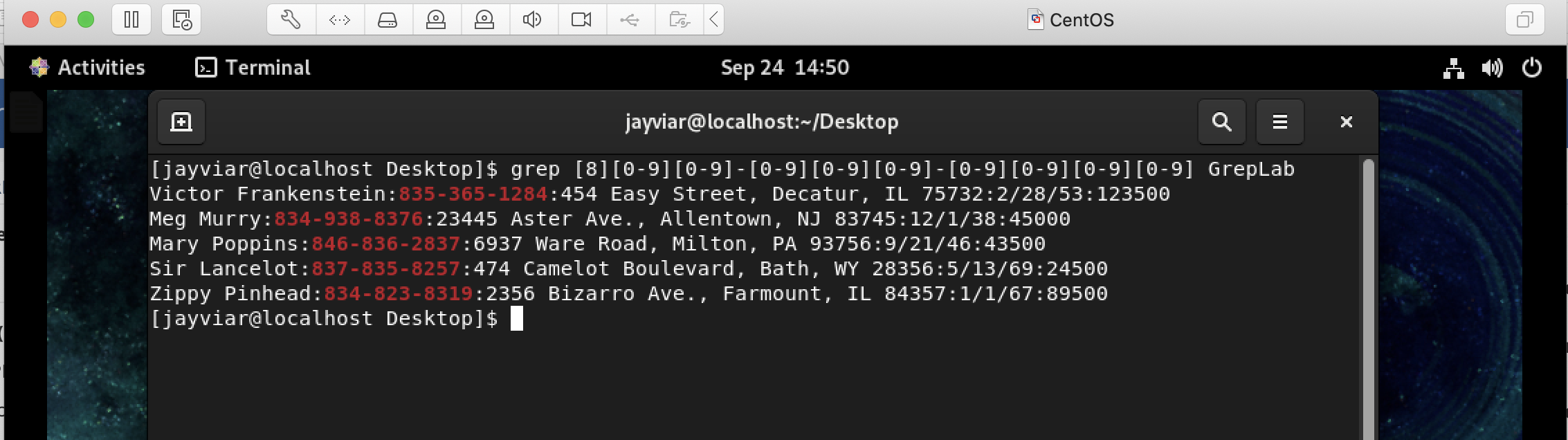
When using grep there are sometimes multiple solutions to find the same answer. For this problem, we know the date format goes like this mm/dd/yy so we can use this to our advantage. We know the month and date will never be 35. So, with this knowledge, we know that searching for /35 will only result in a birth date of the year 1935.



1. Print all lines where the phone number is in an area code that starts with an 8

grep [8][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9] GrepLab

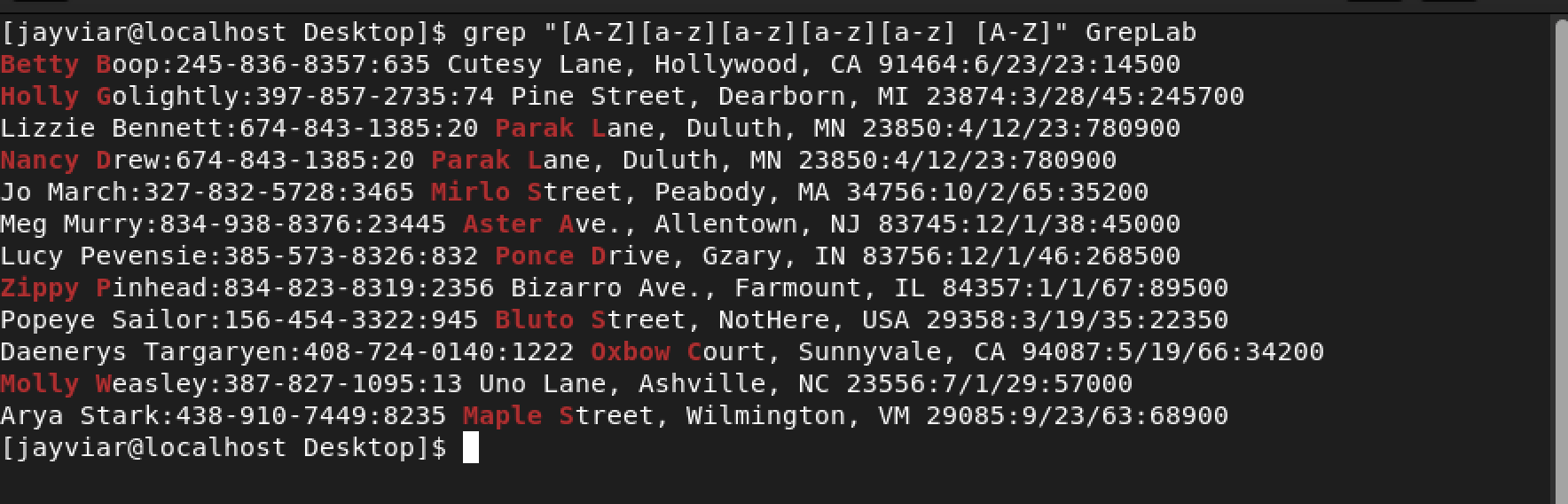
Although there might be a better way to do this. The most standard way of getting this solution is to do it this way. We know that American phone numbers all have the same format. So, we start off by searching the number 8 and then proceed to follow a specific syntax that only phone numbers contain. We use [0-9] as it does not matter what numbers are after but the number 8, as long as we start with it.



1. Print all lines containing an uppercase letter, followed by 4 lowercase letters, a space and one uppercase letter.

grep “[A-Z][a-z][a-z][a-z][a-z] [A-Z]” GrepLab

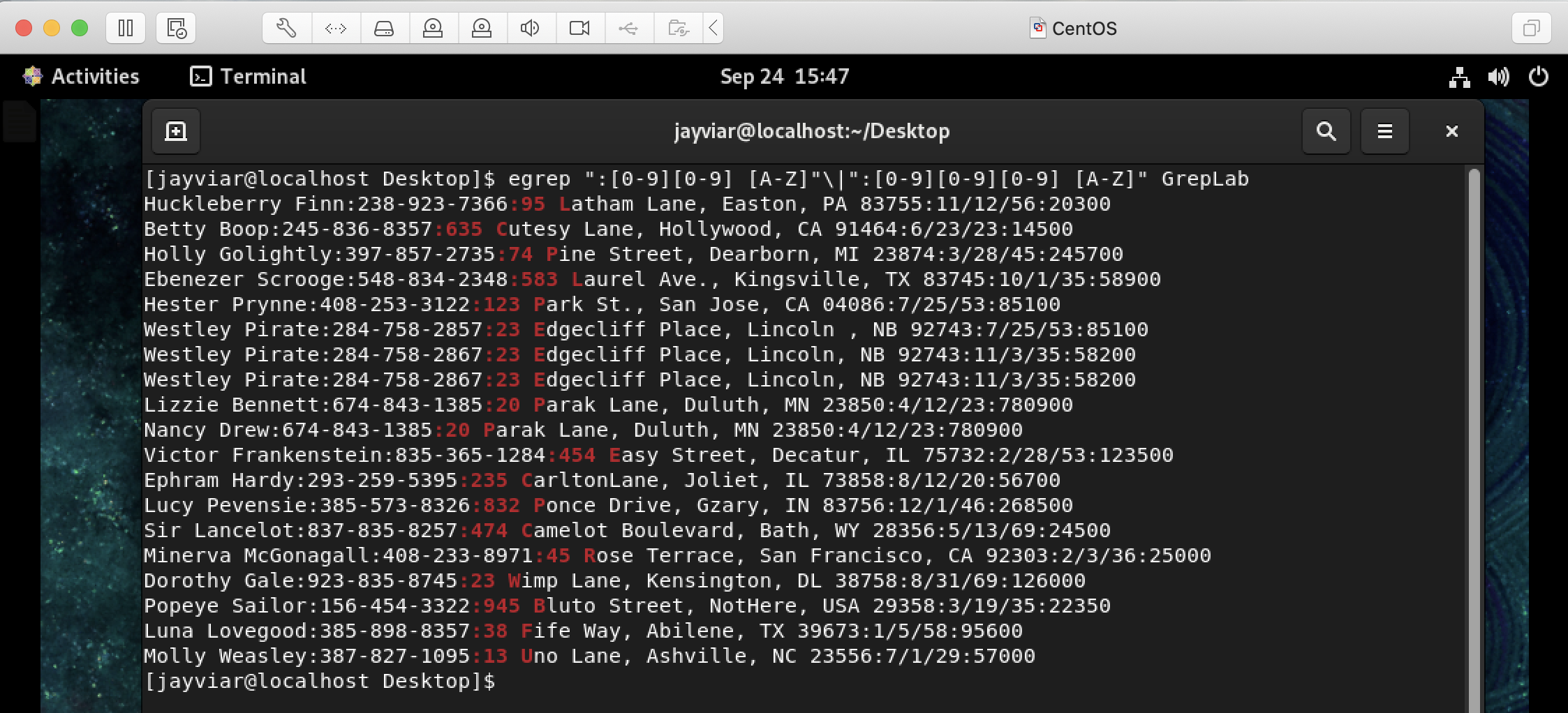
For this Solution, we’re simply finding a string of characters that can be anything but have to start with a capital then 4 lowercase, a space then another capital. We specify capitals. by typing the A-Z range in all caps, we then type 4 consecutive lowercase a-z ranges and include a space and one last Capital A-Z range



1. Print lines where the address begins with a two or three digit number (so this would be 12 main st or 123 main street but not 1234 main street).

egrep “:[0-9][0-9] [A-Z]\|:[0-9][0-9][0-9] [A-Z] GrepLab

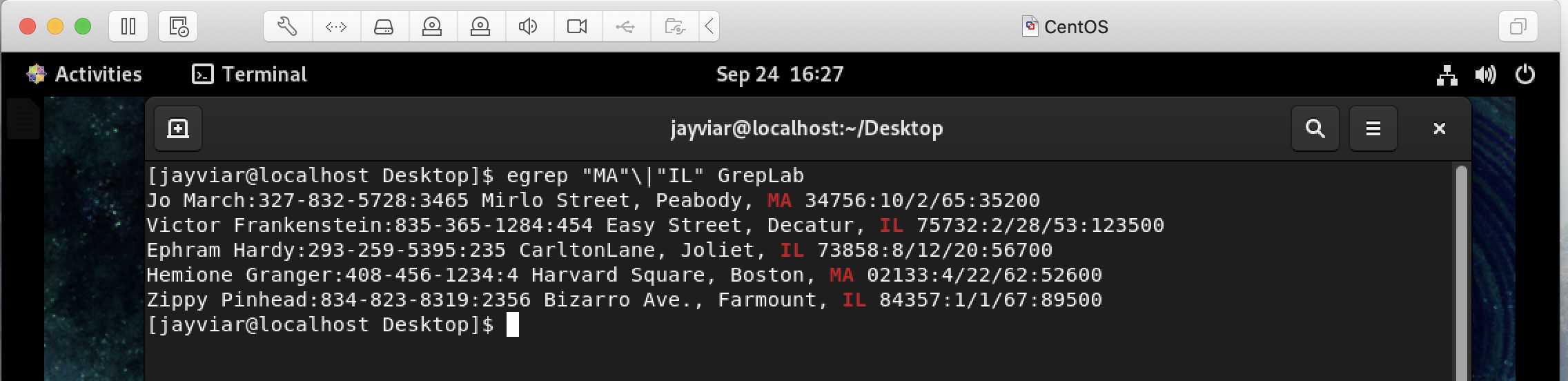
To breakdown this code. We will start off with what we know. We know that every Address will contain a “:” before the address. Again, using what we know to our advantage we can search for this with grep. However, we will be using egrep as we are incorporating extended regular epressions with our code. Moving along, we only want addresses with 2 numbers it as well. In this case we will use [0-9][0-9] two times as we want a 2 digit number. We then know that there is a space between an address number and name, so we include that as well. And finally we will use [A-Z] because house names start with a capital. However, we also want to find addresses with 3 numbers as well. Not a problem for us. We will now introduce the OR command which can look a little like this \|. This command will look for one or the other or both in conjunction with egrep. Then we will type the same code with an extra [0-9] so we can have a 3 digit address.



1. Print lines where the person lives in Mass or Illinois

egrep “MA”\|”IL” GrepLab

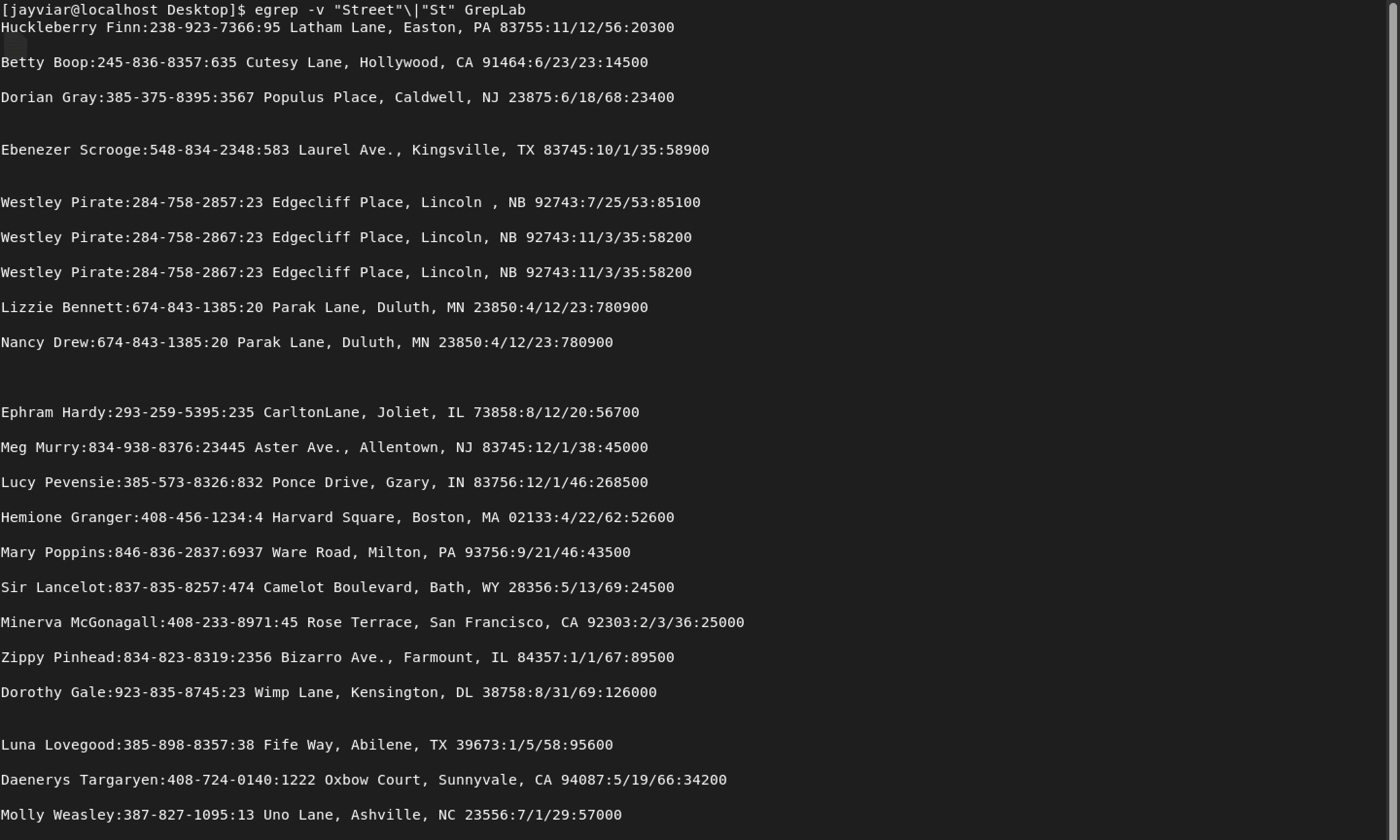
This solution is relatively simple. We start off using our egrep command this time because we are using the pipe command as a part of our regular expression. We then specify the state using uppercase MA because we know the states are filtered by their abbreviated name. We then use our OR command as we want the state of Illinois and for that we type uppercase IL after the or command.



1. Print lines containing the addresses that aren’t on a street (You might see St as shorthand for street)

egrep –v “Street”\|”St” GrepLab

For this final solution. We will be using egrep, -v, and the \| command to search for addresses that are not on a Street or St. We are using egrep to incorparate regular expressions in our search pattern and –v to find the NOT of what we are typing. With both of these combined as well as specifying Street and St with the \| command. We end up finding the results of everything that is not on a street.



Sources-

<https://regex101.com>

<https://www.aholdengouveia.name/LinuxAdmin/Grep.html>

<https://www.aholdengouveia.name/LinuxAdmin/Regex.html>