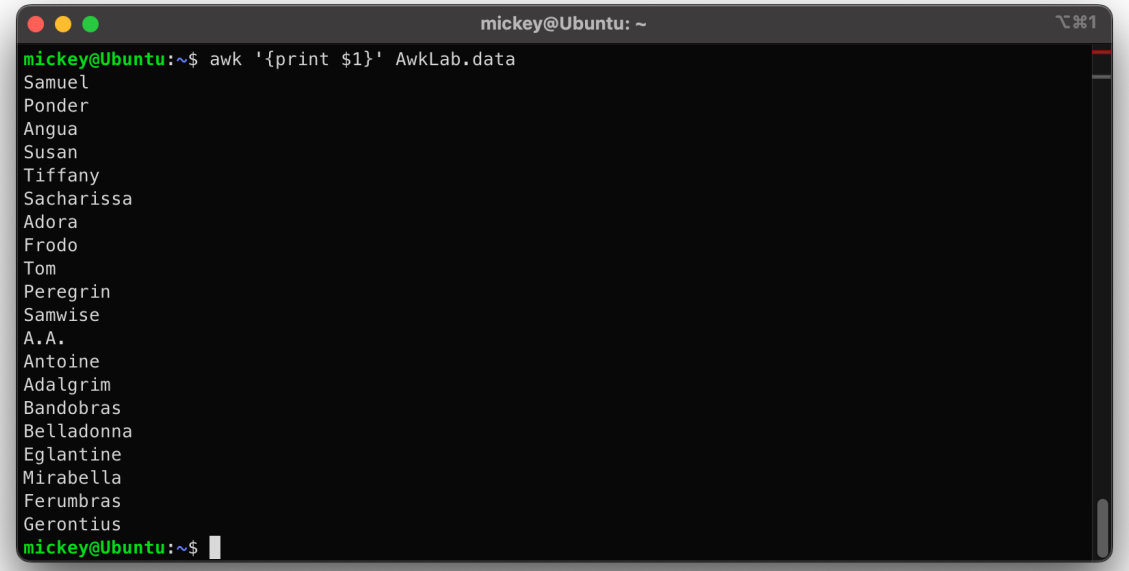
AWK Command

1. Print all the First Names.

Command

```
awk '{print $1}' AwkLab.data
```

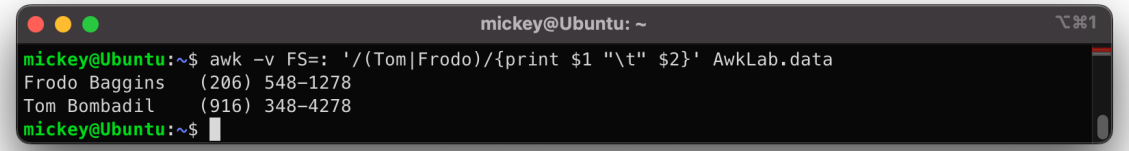


Explanation: By default `awk` uses spaces as the delimiter and we can get everything upto the first space (which in here is the `first name`).

2. Print phone numbers for Tom and Frodo after their names

Command

```
awk -v FS=: '/(Tom|Frodo)/{print $1 "\t" $2}' AwkLab.data
```

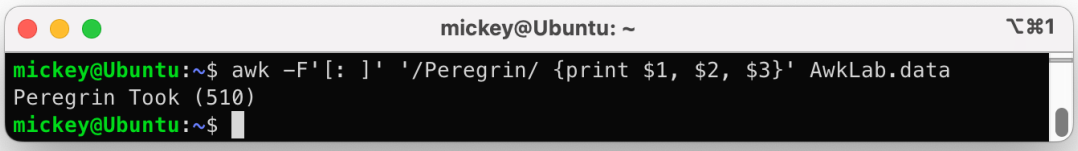


Explanation: Given the data format, to print phone numbers it is better to change the default delimeter `space` to `:` which separates the lines based off the `:` delimiter. To achieve the same, we have to pass the arguement `FS=:`. To reduce the search space to just `Tom` or `Frodo`, we use the regex `(Tom|Frodo)` and print the name and the phone number represented by `$1` and `$2` repectively.

3. Print Peregrin's full name and phone number area code only.

Command

```
awk -F'[: ]' '/Peregrin/ {print $1, $2, $3}' AwkLab.data
```

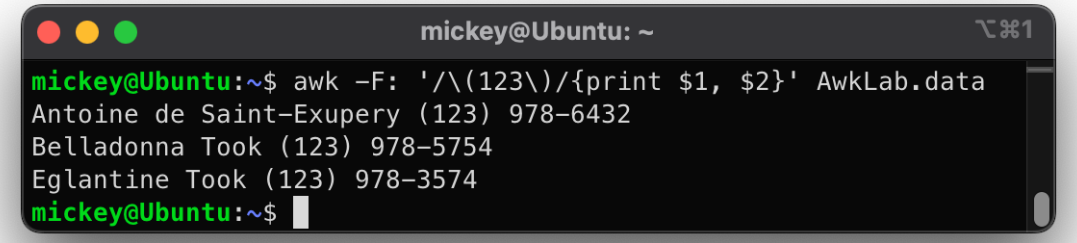


Explanation: Given the data format, it will be best to break the lines based on both `space` and `:` as the delimeters. To achieve the same, we pass the arguement flag `-F'[: ]'`. Using the regex `/Peregrin/` we reduce the search space for `Peregin` only and print the first name, last name and area code represented by `$1`, `$2` and `$3` respectively.

4. Print all phone numbers (full number) in the 123 area code along with the names

Command

```
awk -F: '/\(123\)/{print $1, $2}' AwkLab.data
```

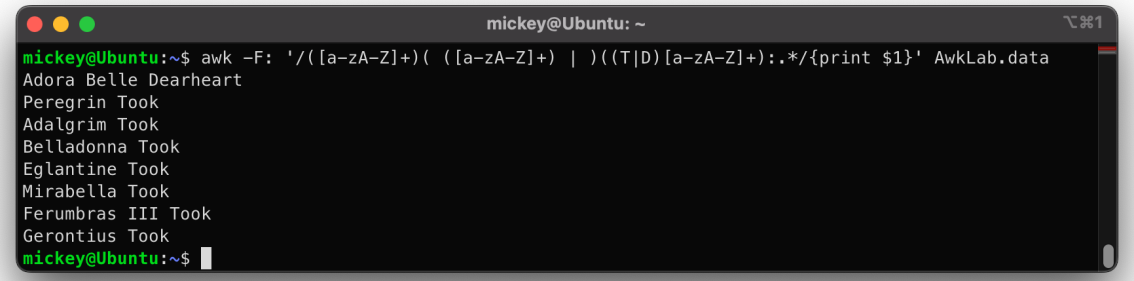


Explanation: Given the data format, to print phone numbers it is better to change the default delimeter `space` to `:` which separates the lines based off the `:` delimiter. To achieve the same, we have to pass the arguement `-F:`. To reduce the search space to `(123)` area code, we use the regex `/\(123\)/` to match the area-code. We can print the full name and complete phone number using `$1` and `$2` accordingly.

5. Print all Last names beginning with either a T or D (careful of middle names!)

Command

```
awk -F'[: ]' '/([a-zA-Z]+)( ([a-zA-Z]+) | )((T|D)[a-zA-Z]+):.*/{print $1}' AwkLab.data
```

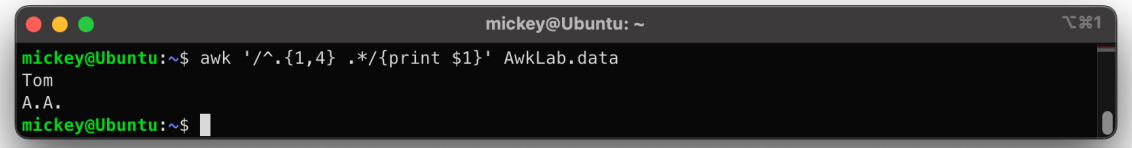


Explanation: Given the data format, to print phone numbers it is better to change the default delimeter `space` to `:` which separates the lines based off the `:` delimiter. To achieve the same, we have to pass the arguement `-F:`. To reduce the search space to last names starting with either `T` or `D` we use the regex : `/([a-zA-Z]+)( ([a-zA-Z]+) | )((T|D)[a-zA-Z]+):.*/`.

6. Print all first names containing four or less characters.

Command

```
awk '/^.{1,4} .*/{print $1}' AwkLab.data
```

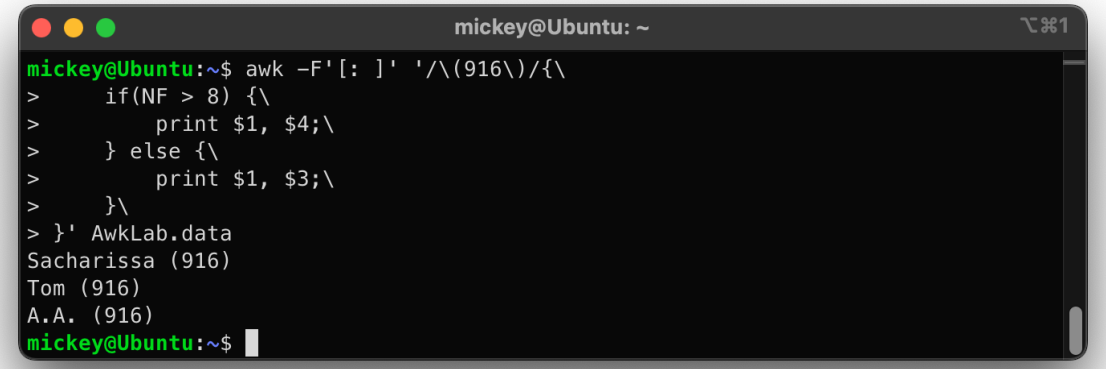


Explanation: To reduce the search space to all the first names containing four or less, we can use the regex `/^.{1,4} .*` which looks for first names of length `1 to 4`. The period (`.`) represents any character and `{1,4}` represents the length of the pattern. So here we are looking for any string of length 1 to 4 with a follwing space and any characters afterwards. We can use the default delimiter and print the first component (i.e. the first name) with `$1`.

7. Print the first names and area codes of all those in the 916 area code.

Command

```
awk -F'[: ]' '/\(916\)/{\
    if(NF > 8) {\
        print $1, $4;\
    } else {\
        print $1, $3;\
    }\
}' AwkLab.data
```

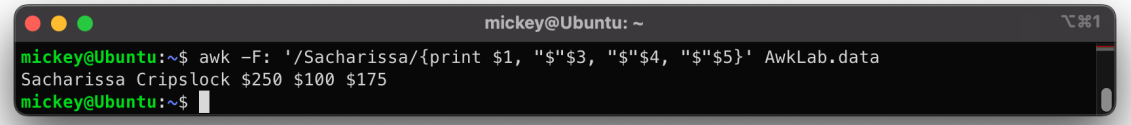


Explanation: To reduce the search space to lines containing the area code 916 we can use the regex `/\(916\)/` and use both space and colon as the delimiters (by passing the arguement `-F'[: ]'`). We can print the first name with `$1` and area code with `$3` or `$4` depending if the name contains a middle name. We can check the presence of a middle name by checking the total number of components. If is greater than 8, we can be sure that the name contains a middle name.

8. Print Sacharissa's campaign contributions following her name. Each value should be printed with a leading dollar sign; e.g., $250 $100 $175.

Command

```
awk -F: '/Sacharissa/{print $1, "$"$3, "$"$4, "$"$5}' AwkLab.data
```



Explanation: Use colon as the delimiter and reduce the search space using the regex `/Sacharissa/`. Print the details accordingly and use the `"$"` for printing inlined characters.

9. Print last names followed by a comma and the phone number. Be careful of the last names's format.

Command

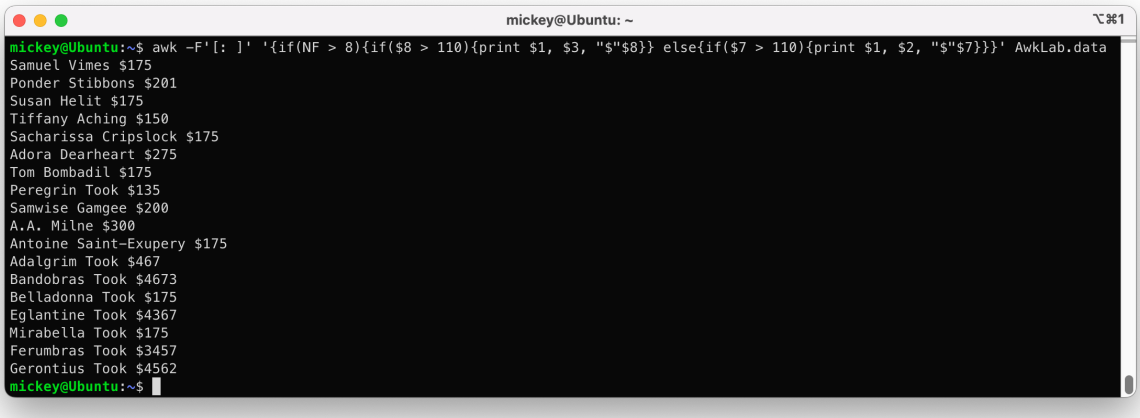
```
awk -F'[: ]' '{if(NF > 8){print $3, $4" "$5;} else{print $2, $3" "$4}}' AwkLab.data
```

Explanation: Use both `colon` and `space` as the delimiters. Check for the presence of middle name. If middle name is present, print last name using $3 else print last name using $2. The area code and phone number follows.

10. Print the first and last names of those who contributed more than $110 in the last month. Make sure to include their last month contribution

Command

```
awk -F'[: ]' '{if(NF > 8){\
    if($8 > 110){\
        print $1, $3, "$"$8;\
    }\
    } else {\
        if($7 > 110) {\
            print $1, $2, "$"$7\
        }\
    }\
    }' AwkLab.data
```

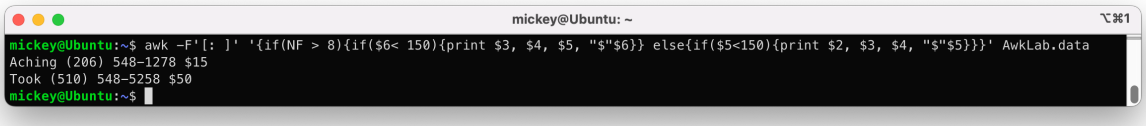


Explanation: Use both `colon` and `space` as the delimiters. Check for the presence of middle name. If middle name is present, print last name using $3 and last month's contribution using $8 else print last name using $2 last month's contribution using $7. In between, we need to check if the last month's contribution is more than `$110`. If not, we will not print anything.

11. Print the last names, phone numbers, and first month contribution of those who contributed less than $150 in the first month.

Command

```
awk -F'[: ]' '{\
    if(NF > 8) {\
        if($6< 150) {\
            print $3, $4, $5, "$"$6\
        }\
    } else {\
        if($5<150) {\
            print $2, $3, $4, "$"$5\
        }\
    }\
}' AwkLab.data
```

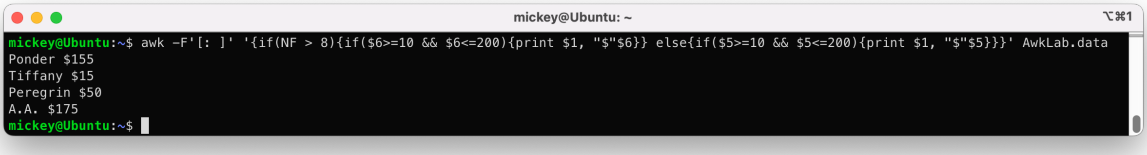


Explanation: Use both `colon` and `space` as the delimiters. Check for the presence of middle name. If middle name is present, print last name using $3, phone number using $4 and $5, and first month's contribution using $6 else print last name using $2, phone number using $3 and `$4`, and first month's contribution using $. In between, we need to check if the first month's contribution is less than `$150` using the `if` statement. If not, we will not print anything.

12. Print the first names and contribution of those who contributed between $10 and $200 in the first month.

Command

```
awk -F'[: ]' '{\
    if(NF > 8) {\
        if($6>=10 && $6<=200) {\
            print $1, "$"$6\
        }\
    } else {\
        if($5>=10 && $5<=200) {\
            print $1, "$"$5\
        }\
    }\
}' AwkLab.data
```

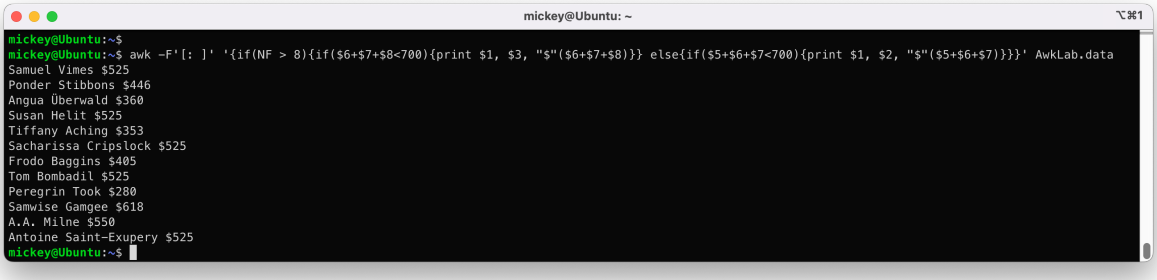


Explanation: Use both `colon` and `space` as the delimiters. Check for the presence of middle name. check for the first month's contribution if it is between `[$10, $200]` and print it accordingly (either using `$5` or `$6`).

13. Print the first name, last names and total contributions of those who contributed less than $700 over the three-month period.

Command

```
awk -F'[: ]' '{\
    if(NF > 8) {\
        if($6+$7+$8<700) {\
            print $1, $3, "$"($6+$7+$8)\
        }\
    } else {\
        if($5+$6+$7<700) {\
            print $1, $2, "$"($5+$6+$7)\
        }\
    }\
}' AwkLab.data
```

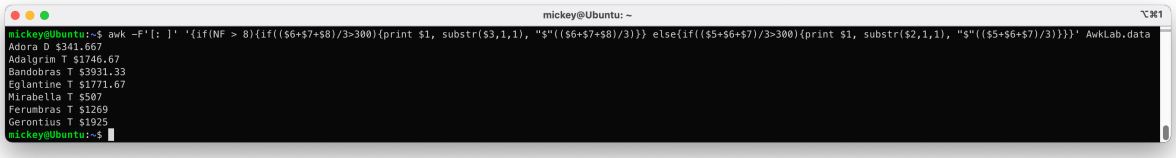


Explanation: Use both `colon` and `space` as the delimiters. Check for the presence of middle name. check for the total contribution if it is less than `$700` and print it accordingly (either using `$6+$7+$8` or`$5+$6+$7`) along with the first name and last name.

14. Print the first names and first letter of the last name, and average contribution of those who had an average contribution of more then $300

Command

```
awk -F'[: ]' '{\
    if(NF > 8) {\
        if(($6+$7+$8)/3>300) {\
            print $1, substr($3,1,1), "$"(($6+$7+$8)/3)\
        }\
    } else {\
        if(($5+$6+$7)/3>300) {\
            print $1, substr($2,1,1), "$"(($5+$6+$7)/3)\
        }\
    }\
}' AwkLab.data
```

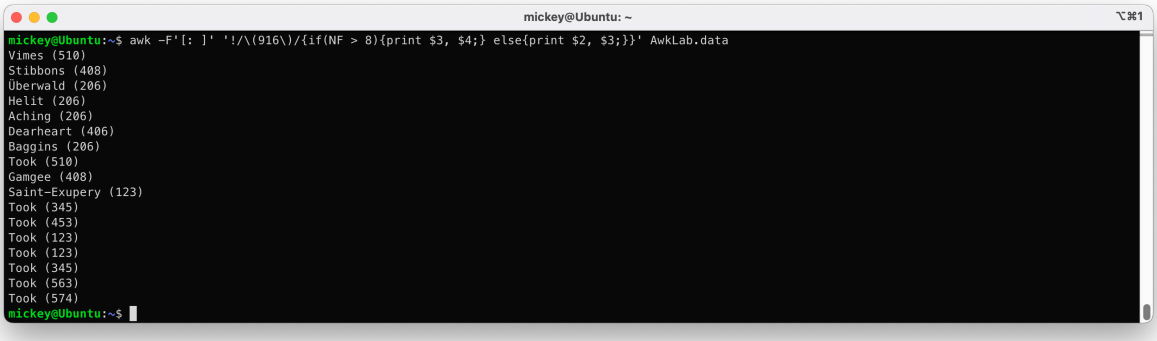


Explanation: Use both `colon` and `space` as the delimiters. Check for the presence of middle name. check for the average contribution (by dividing the total contribution by 3) if it is greater than `$300` and print it accordingly (either using `$6+$7+$8` or`$5+$6+$7`) along with the first name and print use `substr($X,1,1)` to get the first character of the last name where `$X` is the string.

15. Print the last name and area code of those not in the 916 area code.

Command

```
awk -F'[: ]' '!/\(916\)/{\
    if(NF > 8) {\
        print $3, $4;\
    } else {\
        print $2, $3;\
    }\
}' AwkLab.data
```



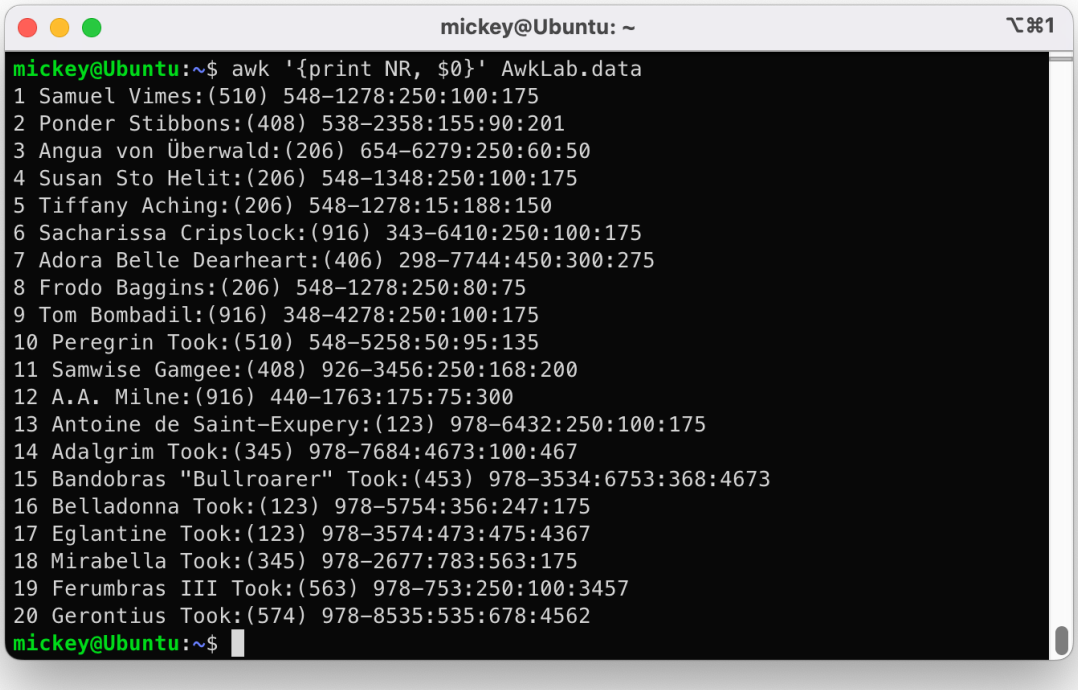
Explanation: To reduce the search space to strings not containing `/\(916\)/` use the `!` before

the regex which tell us the to inverse match the space. Check for the presence of middle name and print accordingly.

16. Print each record preceded by the number of the record.

Command

```
awk '{print NR, $0}' AwkLab.data
```

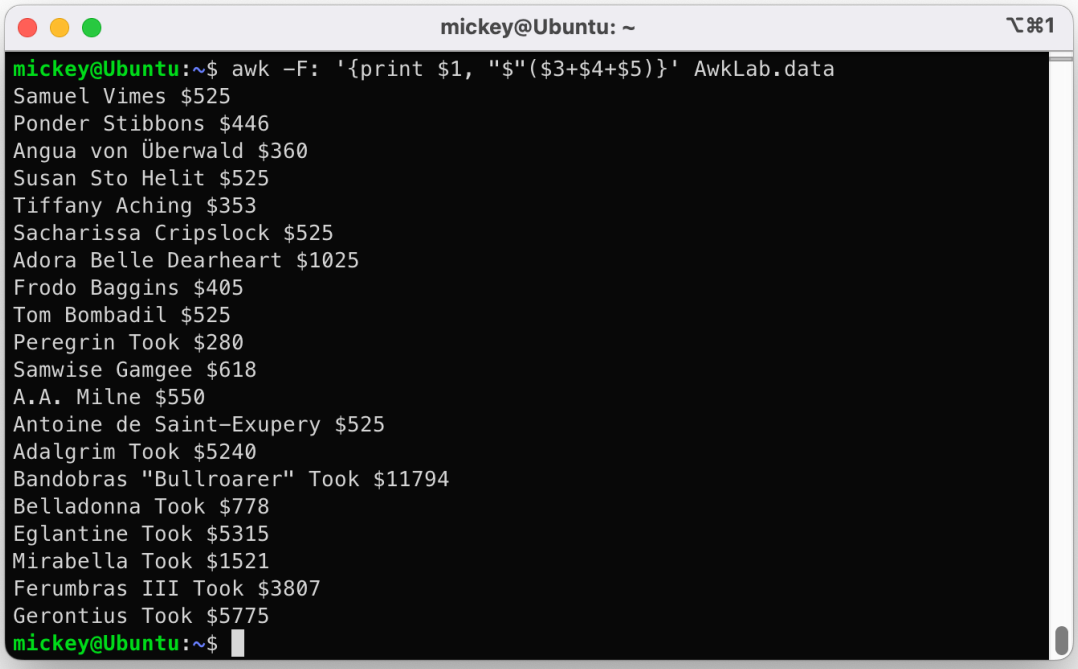


Explanation: "NR" is a special built-in variable of AWK that stands for "number of records". It keeps count of the number of line and we can print the NR variable to get the current line and print it accordingly.

17. Print the name and total contribution of each person.

Command

```
awk -F: '{print $1, "$"($3+$4+$5)}' AwkLab.data
```

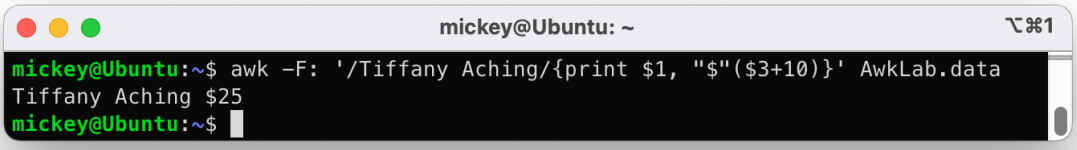


Explanation: Using colon as the only delimeter, we can print the full name represented by `$1` and add up `$3+$4+$5` to get the total contribution.

18. Add $10 to Tiffany Aching's first contribution and print her full name and first contribution.

Command

```
awk -F: '/Tiffany Aching/{print $1, "$"($3+10)}' AwkLab.data
```

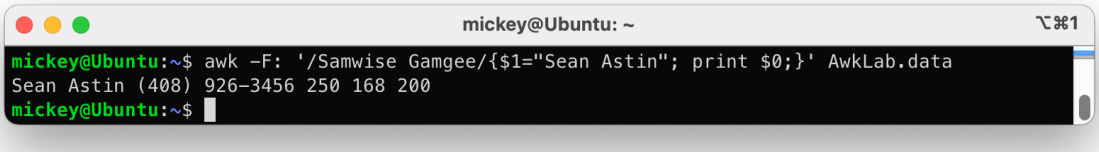


Explanation: Reduce the search space using the regex `/Tiffany Aching/`. Using the colon delimeter, we can get the full name and print the first contribution using `$3` after adding `10` to the value.

19. Change Samwise Gamgee's name to Sean Astin

Command

```
awk -F: '/Samwise Gamgee/{$1="Sean Astin"; print $0;}' AwkLab.data
```



Explanation: Reduce the search space using the regex `/Samwise Gamgee/`. Using the colon delimeter, we can change the var `$1` to `Sean Astin` and print the entire record using `$0`.

20. Write an awk script to do the following (MUST be an awk script not just a bash script or commands on the commandline)

(a) Prints first name of the all the Tooks followed by their total campaign contributions.

(b) Print the full names and contributions of anyone who contributed between $10 and $200 in the last contribution

(c) Prints the full names and average contribution of those who contributed less than $300 on average Command

```
#!/usr/bin/awk -f

BEGIN {FS="[: ]"}{
    if(NF > 8){
        if($3=="Took"){
            print $1, "$"($6+$7+$8);
        }

        if($8 >= 10 && $8 <= 200){
            print $1, $2, $3, "| Last Contribution: $"$8;
        }

        if(($8+$7+$6)/3 < 300){
            print $1, $2, $3, "| Average: $"(($8+$7+$6)/3);
        }

    } else {
        if($2 == "Took"){
            print $1, "$"($5+$6+$7);
        }

        if($7 >= 10 && $7 <= 200){
            print $1, $2, "| Last Contribution: $"$7;
        }

        if(($7+$6+$5)/3 < 300){
            print $1, $2, "| Average: $"(($7+$6+$5)/3);
        }
    }
}
```

```
# Make it executable
chmod +x awk_script.sh

# Run the script
./awk_script.sh AwkLab.data
```

Explanation: NF is a predefined variable whose value is the number of fields in the current record.

The script uses both colon and space as the delimeters and checks for the presence of a middle name by comparing the `NF` variable. Firstly, it checks for the last name if it is `Took` and if true, prints the first name. Then checks for the last contribution and finally checks for average and prints it.

References

1. https://marinegeo.github.io/2018-08-10-adding-images-markdown/
2. https://stackoverflow.com/questions/8257865/can-field-separator-in-awk-encompass-multiple-characters
3. https://www.tecmint.com/write-shell-scripts-in-awk-programming/
4. https://www.unix.com/shell-programming-and-scripting/143768-use-awk-print-first-6-characters.html
5. https://www.aholdengouveia.name/LinuxAdmin/Awk.html
6. https://flylib.com/books/en/4.356.1.52/1/
7. https://unix.stackexchange.com/questions/318846/inverse-regex-in-awk
8. https://linuxhint.com/nr-built-in-awk