**Awk Guide**

**Disclaimer**:

Some Pictures were taken in SmarTTY instead of virtual box as I have figured how much more user friendly it is but you do not need it but it is recommended

**Question 1 Using Awk to get the first name**

The command I use to get the all the first names in the **AwkLab.data** file is **awk ‘{print $1}’ AwkLab.data** as shown below

A screen shot of a computer

Description automatically generated

**Explanation:** Every awk command starts with **awk** then in the quotes put how I want to use the **awk** command. The brackets portion in the can be used to print any portion of the line putting **print** in the square bracket. Without specifying what pattern of characters to look for prior to the brackets it will print every line only specifying what to print in the brackets which is the **$1** which states to print the first portion of the code which by default unless otherwise specified uses a space as delimiter to separate each portion of the line. If you look at file using the grep command, you see the first name is first portion of the code that is separated by spaces.

A screenshot of a computer

Description automatically generated

After the quotes it’s just like any other command put the file name or location of the file you want use your command for and make sure to include the extension if it is part of the name like **.data**

**Question 2 Getting Tom and Frodo’s name and Number**

The command I used to get Tom and Frodo’s name and number is **awk -F: ‘/Tom|Frodo/ {print $1 “\t” $2}’ AwkLab.data** as shown below

A screen shot of a computer

Description automatically generated

**Explanation:** Before explaining the code first let’s look at how each user is formatted in **AwkLab.data** file

A screenshot of a computer

Description automatically generated

By Default **awk** uses space as a delimiter to separate each portion of the code each portion is characterized as **$n** so **$1** is lets say in the picture of the first line **Samuel** as that is the first portion of the line where each portion is separated by the space. Trying to Get Tom and Frodo’s Name and Phone Number can be tricky as if we look at the first line of the picture above **$2** is **Vimes:(510)** as there is no space between the last name, colon and phone number. Thankfully after the **awk** we can use **-F(char)** flag to change the delimiter I choose **:** as that is more used as a delimiter in the file than space so I format as **-F:** now instead of each portion of each line being separated by space it will be separated by a colon so now using the first line from the picture again **$1** is **Samuel Vimes** as it is no longer separated by space.

In the quotes to find a pattern of characters before printing you put what you want in between slashes like this **/text to find/** I am looking for Tom and Frodo to look for multiple things at once we can use the **|** expression which will be treated as an or sign between the **|** symbol so I put in the slashes **/Tom|Frodo/** which will state that the pattern of characters I want to look for is either **Tom** or **Frodo** unlike **sed** or **grep** you need to specify what portion of the code you want displayed once on lines where it has the matching pattern that is what **{print $1 “\t” $2}**  where **print** is to print from the matched line the **$1** is now **Frodo Baggins** as the portion is now separated by a colon instead of space the **“\t”** is a TAB to have a space for the printed output between each part being printed. Since the command now uses a colon as a delimiter the phone number is now **$2** or the second portion of the line as each portion of the phone is either separated by a space or dash so it is now combined into one portion. After the bracket portion but the closing quote the name or location of the file you want to use for this command which is **AwkLab.data**

**Question 3 Getting Peregrin’s Name and Area code**

The command I used to get Peregrin’s Full name and the area code portion of the phone number is **awk -f ‘[:()]’ ‘/Peregrin/ {print $1 “\t” $2} AwkLab.data** as shown below

**A black screen with white text

Description automatically generated**

**Explanation:**  The **-F ‘[char]’** flag can be used to change the delimiter which by default space is used to separate each portion of a line from the file in the square brackets I put **:()** so each line will either be separated by a colon or an opening and closing parentheses so anything in a line that does not have either of those is it’s own portion. In the quotes in the slash I part I put the pattern of characters I’m looking for which is **/Peregrin/** in the print portion of the code with the brackets I print the first portion **$1** which is the full name of Peregrin a **“\t”** to have a space in between each portion of the line from the print the third portion **$3** which is the area code of the phone number which is now its own portion separated by the parentheses. Then do the closing quote and the exact name or location of the file extension included if it is part of the name so **AwkLab.data**.

**Question 4 Getting Users with an area code of 123**

The command I used to get every user with the area code of **123** was **awk -F: ‘/(123)/ {print $1 “\t” $2} AwkLab.data** as shown below

A screenshot of a computer

Description automatically generated

**Explanation:** The **-F:** flag is used to have each portion of the line of the file be separated by a colon instead of a space this makes the full name and phone numbers their own variables instead of each portion of those two being separate variables separated by space. In quotes in the slash section, I put the pattern of characters I am looking for the phone number area code is wrapped in parentheses giving it a unique pattern so I put **(123)** in the slashes part in the print portions in the bracket I **print** the first portion **$1** which is the full name a **“\t”** which gives a space between the output of the portions and the second portion which is the **$2** after the closing bracket and quote I put the name or location the file I want to modify which is **AwkLab.data**.

**Question 5 Getting Users who Last name start with T or D**

The command I used to get every user with a last name starting in T or D is **awk -F: '$1 ~ / [TD]/ {split($1, name, " "); if (length(name) >= 3) print name[length(name)]; else print name[2];}' AwkLab.data** as shown below.

A computer screen with white text

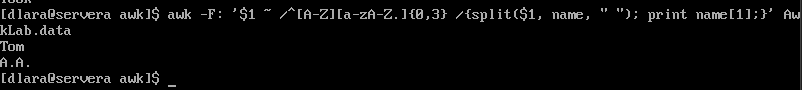
Description automatically generated

**Explanation:** The **-F:** flag changes the delimiter from a space to a colon so now each portion of a line is separated by a colon instead of a space in the quotes I put the first portion of the line **$1** which is each users full name the **~** followed by the slashes show what to match from the first portion of the code which is **/ [TD]/** a space which what is followed from the first name within the portion then **[TD]** which since each last name is Capitalized in the first letter it will always match the first letter and the **T** and **D** which will find the letter either **T** or **D** after the slashes part

To ensure I only print the last name I can print only a portion of the segment **$1** the first segment of the line with **{split($1, name, “ “};** where **$1** is the portion I am creating an array from **name**  is the reference and **“ “** is the delimiter I can simply choose to print from a part of the array in theory I should now print **name[2]** but I need to account for middle name so I set an if statement **if (length(name) >= 3)** a length of the array is determined by how many segments separated by delimiter a full name without its middle name should have two segments where **name[1]** is the first name and **name[2]** therefore having a length of 2 however a full name that includes it’s middle name will have length of 3 so if the length of the array is 3 or greater print **name[length(name)]** sin which will print the number of the length of the array which is where the last name will be else **print name[2]** since it the second part is not only the max length but where the last name will always be after a closing **;}** then **‘** put the name of the file that is being used which is **AwkLab.Data**

**Question 6 Getting Users who’s first name is 4 letters or less**

The command I used to get users who’s first name is four letters or less is **awk -F: ‘$1 ~ /[A-Z][a-zA-Z.]{0,3}/{print $1}’ AwkLab.data** as shown below



**Explanation:** The **-F:** flag is used to have each portion of a line in the file be separated by a colon instead of a space, so the full name is one variable as each part of the name is separated by spaces instead of colon. In the quotes I put **$1** for the first portion of the line which is the full name a **~** followed by the slashes show what to match from the first portion of the code which is **/^[A-Z][a-zA-Z.]{0,3} /** this uses multiple expressions and anchors the **^** tells specifies to look for the pattern of character now specified from the **$1** portion of the line as specified previously as this is about the first name of the code the **[A-Z]** is an expression to find any uppercase letter as the first letter is uppercased the second expression **[a-zA-Z.]** accounts for both uppercase and lower case letters in the name letters along with any periods between names the **{0,3}** expression tells to find  **0** to **3**  simultaneous instances of the previous expression in this position of the pattern and a space as that is what is followed after the first name in the pattern. To print only the first name I can separate the b **$1** into any array once the pattern is matched that is what **{split{$1, name, “ “) ;** where **split** is to split the **$1** is what instance am I splitting which is the full name the **name** is the name of the array and the **“ “** to use as space as a delimiter and a **;** to close out this portion of the code. I then print the first part of the array which is just the first name so **print name[1]** to only print first name as that is the first part of the **name** array from the **$1** portion of the code previously separated by a colon. After a **;}** to close out that portion of the code and then a closing quote then the name or location of the file to modify which is **AwkLab.data**

**Question 7 Getting the first names of Users with 916 as the area code**

The command I used to get the first name and area code of users with an area code of 916 is **awk -F: ‘/(916)/{split($1, name, “ “); split($2, num, “ “); print name[1] “ ” num[1];’ AwkLab.data** as shown below

A black screen with white text

Description automatically generated

**Explanation:** The **-F:** flag after the **awk** makes a colon the delimiter so awk separates each portion of the line by a colon instead of a space In the quotes I first have **/(916)/** where I put the pattern of characters to match in the slashes which is **(916)** as area codes in the file is wrapped in parentheses. To only print portions of the code I can create arrays from separate portions of the line using the **split** command to create arrays **{split($1, name, “ “);** creates an array from the first portion of a line **$1** which is the full name **name** is the name of the array and **“ “** uses a space as a delimiter as each part of the name is separated by a space I also create an array from the phone number **split($2, num, “ “);** where **$2** is the second portion of the code which is the phone number **num** is the name of the array and **“ “** uses a space as a delimiter for this array as the area code part of the number is separated by a space instead of a dash make sure to put semi colons after each split statement. Afterwards I simply **print** I first print the first name portion of the **name[]** array which is **name[1]** then a **“ ”** to make a space between the two portions to print the **num[1]** to print the area code portion of the code which is the first part of the **num[]** array I then putting a closing **;}** then a closing part then the name or location of the file to use this command on which **AwkLab.data**

**Question 8 Getting Sacharissa’s Contributions then name**

The command I used to get Sacharissa’s Contributions then full name was **awk -F: ‘/Sacharissa / {print “$” $3, “$” $4, “$” $5, $1}’ testfile** as shown below



**Explanation:** The **-F:** flag after the **awk** has it so a colon is used as a delimiter instead of a space this is useful as each portion of a line is bordered by a colon so each field is separated by a colon. In the quotes in between the slashes I put the pattern of characters to match which is **Sacharissa** followed by space as that is what proceeds the first name In the brackets portion I put what I want to **print** which is the contributions with a dollar sign next to it follow by Sacharissa’s name so In each quote portion I put the string to output then outside the fields so **“$”** then third field **$3** for the first month contributions followed by a **,** for space instead having to do **“ “** each time and repeat for the fourth and fifth field **$4 $5** for the second and third contributions followed by commas for space then the first field **$1** for the Sacharissa’s full name then outside the closing quote the file to search through which **testfile**. If you look at the other explanations, you would see I usually use the **AwkLab.data** to use my commands however there is problem the **AwkLab.data** file may be bugged as when I try to do the following command on that file I get the following output



Additionally playing around and printing the fields in different orders and messing around with the prints gets the following outputs











Theoretically speaking I should be able to do the command I did for question 8 on the **AwkLab.data** file but the pictures show some strange behavior and outputs that in theory should not be outputted talking to peers they theorized it could be with the file so I created a separate **nano** file (a text editor in linux) called **testfile** and simply manually inputted the line I was searching from into **testfile**

A screenshot of a computer

Description automatically generated

If you are curious as to how I knew what to put into testfile I simply did the following **grep** command **grep “Sacharissa” AwkLab.data** which would print the whole line with the **Sacharissa** pattern as shown below.

A close up of a phone number

Description automatically generated

**I** did not do a copy of the file or append the output from a command that would get the line into **testfile** as that could bring along the same potential file bugs or encoding bugs I simply manually inputted the line into **testfile** then ctrl + x to exit then y to confirm save then enter to finish save and exit. I am not sure what is it about the **AwkLab.data** file to cause such strange behavior but at least I know the command is not the problem the file is I have not had any issues on the files end so far.

**Question 9 Printing Last Name comma then Phone Number**

The command I used to print the last name and phone numbers was **awk -F: {split($1, name, “ “); if (length(name) >= 3) print name[length(name)] “, “ $2; else print name[2] “, “ $2:}’ AwkLab.data** as shown below

**A computer screen with white text

Description automatically generated**

**Explanation:** The **-F:** flag after the **awk** uses a colon as a delimiter instead of a space which is how each segment of the line is bordered In the quotes I do multiple things. I first create an array from the first field which is where the first name is **split($1, name, “ “);** where **$1** is the field I am creating the delimiter **name** is the name of the array and the quotes uses a space as delimiter. Each portion of the field is now it’s own field within the array separated from the space in theory the first field should be the first name and the second the last name but some names in the file have middle names which could the second field the middle name. To account for this I set up a condition in **if(length(name) >=) print name[length(name)] “, “ $2;** The length shows how many fields in a array so if an array has 5 fields the length is 5 which would also be the number of the last field the last field in this is also where the last name portion of the array would always be. So with that I can determine that if the length is greater or equal to 3 then print **name[length(name)]** which will print the field of the array size but also the last field where the last name is the **“, “** is used to print a comma after that array then I print the second which is the phone number. The else portion is if the situation I described earlier regarding middle names is not the case so I print what I did last time except I print the second field in the **name[]** array since that is where the last name will always be. After the closing bracket and quote put the name of the file I want to use the command on which is **AwkLab.data**

**Question 10 Printing Users who’s last contribution was Greater Than a 110**

The command I used to print users who’s last monthly contribution was greater than a 100 was **awk -F ‘{if ($5 > 100) {split($1, name, “ “) print name[1], name[length(name)] name “$” $5}’ AwkLab.data** as show below

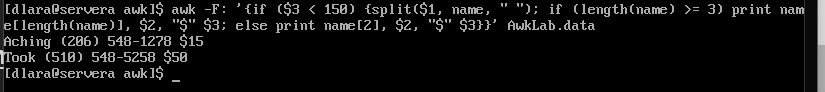
A screen shot of a computer

Description automatically generated

**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space which is important since each monthly contribution of a line is separated a by a colon so each of those contributions are a field. In the quotes I do two things the first is set a condition **if (int($5) > 100)** that if the fifth field which is the last contribution is greater than 100 I do the following which is **print name[1], name[length(name)] $5** which prints the first field of the **name[]** array which is where the first name is the **name[length(name)]** which prints the last field of the array which is where the last name as the number of the length of the array is the number of the last field of the array then a comma for a space a “$” for a dollar sign to printed than the fifth field for the contribution outside the quotes and brackets I put the name of the file I want to modify which is **AwkLab.data**

**Question 11 Getting the last name, phone number and first monthly contribution less than 150**

The command I used to get the last name, phone number and first monthly contribution less than 150 was **awk -F: ‘{if ($3 < 150) {split($1, name, “ “); if (length(name) >= 3) print name[length(name)], $2, “$” $3; else print name [2], $2, “$” $3}}’ AwkLab.data** as show below



**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space space which is important since each monthly contribution of a line is separated a by a colon so each of those contributions are a field. In the quotes I set a condition **{if ($3 <150)** that if the 3rd field **$3** which is where the first monthly is under the delimiter rules set prior then I first create an array from the first field **{split($1, name, “ “);** where **$1** is where I’m creating the array from **name** is the name of the array and **“ “** has a space used as delimiter as each portion of the name is separated by a space. To account for middle names since those within the array change where the last name is I set another condition after **if (length(name) >= 3)** an array with just the first and last name will have the length of 2 for the first field as the first name and the second field as the last name so if it is 3 or greater I do the following. **Print name[length(name)], $2, “$” $3;** which prints name from the array position of the number of the length of the array which is not only the size of the number but the number of the last field of the array to account with name fields with middle names on them then a comma for a space the second field **$2** for the phone number followed by comma for the space then a **“$”** for a dollar sign then the third field **$3** for the contribution I then do an else for the full name fields where there is only a first and last name which prints the same thing except for the name array I do **name[2]** since that is where the last name would otherwise be in the array. After the closing brackets and quote I do the name of the file I want to modify which is **AwkLab.data**

**Question 12 Printing Users who’s first contribution was**

**Between 10 and 200**

The command I used to print Users who’s first contribution was Between 10 and 200 was **awk -F: ‘{if ($3 > 10 && $3 < 200) {split($1, name, “ “); name[1]. “$” $3}}’** **AwkLab.data** as shown below

A screenshot of a cell phone

Description automatically generated

**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space space which is important since each monthly contribution of a line is separated a by a colon so each of those contributions are a field. In the quotes I set the condition **if ($3 > 10 && $3 < 200)** that if the 3rd field **$3** where the first contribution is greater than 10 and **&&** the third field is less than 200 then **{split($1, name, “ “);** which creates an array called **name** under the split command which comes from the first field **$1** and uses a space as a delimiter **“ “** then I print **name[1]** the first name of the user which is located in the first field of the array a comma a **“$”** to print a dollar sign then the third field which is the contribution after the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 13 Getting the Users who’s Overall contribution was less than 700**

The command I used to get Users who’s overall contribution was less than 700 was **awk -F: '{sum = $3 + $4 + $5} { if (sum < 700){ split($1, name, " "); print name[1], name[length(name)], "$" sum}}' AwkLab.data** as show below

A computer screen with white text

Description automatically generated

(picture taken in SmarTTY)

**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space space which is important since each monthly contribution of a line is separated a by a colon so each of those contributions are a field. In the quotes I can create a variable called sum in brackets with **{sum = $3 + $4 + $5}** which makes a variable called sum that adds the 3rd, 4th and 5th field **$3, $4 $5** which are the 1st – 3rd contributions added together I then set condition **if (sum < 700)** that if the variable **sum** is less than 700 than first create an array with the split function **split($1, name, " ");** which creates an array from the first field which is where the full name is called **name** which uses a space as a delimiter I then print **name[1]** for the first name located in the first field of the array then a comma for a space then **name[length(name)]** which prints name from the array position of the number of the length of the array which is not only the size of the number but the number of the last field of the array to account with name fields with middle names on them then a comma for a space then **“$”** to print a dollar sign then **sum** which is the total contributions. after the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 14 Printing First name first letter of last name of users who averaged more than 300 on their contributions**

The command I used to print the First name first letter of last name of users who averaged more than 300 on their contributions was **awk -F: '{avg = $3 + $4 + $5 / 3} {if (avg > 300) { split ($1, name, " "); print name[1], substr(name[length(name)], 1, 1), avg}}' AwkLab.data** as shown below

A screen shot of a computer

Description automatically generated

**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space which is important since each monthly contribution of a line is separated a by a colon so each of those contributions are a field In the quotes I can create a variable called **avg** in brackets with **{avg = $3 + $4 + $5 / 3}** which calculates the average of the 3 contributions from 3rd – 5th fields **$3, $4** and **$5**. I then set a condition in **if (avg > 300)** that if the variable **avg** is more than 300 then I first create an array with the split function **split ($1, name, " ");**  which creates an array from the first field which is where the full name is called **name** which uses a space as a delimiter I then print I then print **name[1]** for the first name located in the first field of the array then a comma for a space then to print a portion of a portion of an array I use the **substr** method on **name[length(name)]** which prints name from the array position of the number of the length of the array which is not only the size of the number but the number of the last field of the array to account with name fields with middle names the next portions are where to print from so from the first character **1,** to the first **1** printing only the first character then a comma for a space a **“$”** to print a dollar sign then **avg** for the average. After the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 15 Printing Users who do not have the area code 916**

The command I used to print users who do not have the area code 916 is **awk -F: '!/(916)/ {split($1, name, " "); split($2, num, " "); print name[length(name)], num[1]}' AwkLab.data** as shown below

A computer screen with white text

Description automatically generated

**Explanation** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space. In the quotes I would in the slashes put the pattern of characters to match but since this is about filtering out lines with a pattern I put the **!** command before the slashes to show to match patterns not containing the pattern to match which is **(916)** followed by a space since the area code is proceeded by one. I then create an array with the split function **split ($1, name, " ");**  which creates an array from the first field which is where the full name is called **name** which uses a space as a delimiter I also create another array **split($2, num, " ");** which uses the 2nd field **$2** which is the phone number calls the array **num** then uses a space as a delimiter since the area code portion is separated by a space. I then print the last name with **name[length(name)]** which prints name from the array position of the number of the length of the array which is not only the size of the number but the number of the last field of the array to account with name fields with middle names. Then a comma for a space then **num[1]** to print the first portion of the **num[]** array which is where the area code portion is. After the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 16 Printing The Record Number then Record**

The command I used to Print each record preceded by the number of the record. Is **awk ‘{print NR, $0}’ AwkLab.data** as shown below

A screenshot of a computer

Description automatically generated

**Explanation:** In the quotes in the in the brackets I print the following **NR** which is the record number of a file so number of the line and 0 field **$0** prints the whole line. After the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 17 Printing every user’s over contribution**

The command I used to print Every user’s overall Contribution was **awk -F: ‘{totalMoney = $3 + $4 + $5} {print $1, “Overall Money: $” totalMoney}’ AwkLab.data** as shown below

A screenshot of a computer

Description automatically generated

**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space which is important since each monthly contribution of a line is separated a by a colon so each of those contributions are a field. In the quotes I create variable in the first set of brackets called totalMoney **{totalMoney = $3 + $4 +$5}** which adds the 3rd through 5th fields **$3 $4 $5** which is where each of the monthly contributions are set by delimiter prior. I then print the first field which is the first name a comma for a space and the string in the quotes **“Overall Money: $”** to print that for each output then the variable **totalMoney** which prints the total money of each person. After the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 18 Adding $25 to Tiffany’s first contribution**

The command I used to add $25 to Tiffany’s first contribution was **awk -F: ‘/Tiffany / {print $1, $3 + 10}’ AwkLab.data** as shown below:

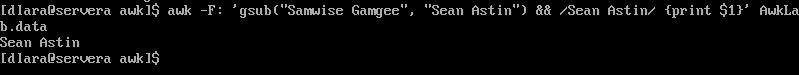
A black screen with white text

Description automatically generated

**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space. In the quotes I first put the pattern of characters to match in the slashes which is **/Tiffany /** after that in the next set of brackets I print **$1** which is the full name a comma for a space the third field **$3** for first contribution and a **+ 10** which adds to ten to that field. After the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 19 Changing Samwise Gamgee to Sean Astin**

The command I use to change Samwise Gamgee to Sean Astin was **awk -F: ‘gsub(“Samwise Gamgee”, “Sean Astin”) && /Sean Astin / {print $1}’ AwkLab.data** as shown below



**Explanation:** The **-F:** flag after the **awk** sets the delimiter to a colon instead of space. In the quotes I can replace a matched pattern to output with a replaced pattern using the gsub function **gsub(“Samwise Gamgee, “Sean Astin”)** working similar **sed** where in the first portion I put the pattern of characters to replace **“Samwise Gamgee”** then in the second portion I put what I want to replace it with which is **“Sean Astin”** I then do **&&** to do multiple commands which is to **print** the first field **$1** which is the full name from the matched pattern printing **Sean Astin**. After the closing brackets and quote I put the name of the file I want to modify which is **AwkLab.data**

**Question 20 Making Awk Script**

The awk Script I used to Get the first name of the tooks along with their total campaign contributions, Printed Users who last contribution was between 10 and 200 and printed users who averaged less than 300 was

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

**{FS = ":"}**

**/ Took/ {**

**split($1, name, " ")**

**firstName = name[1]**

**tookMoney = $3 + $4 + $5**

**print firstName, "$" tookMoney**

**}**

**{**

**if ($5 > 10 && $5 < 200){**

**print $1, "$", $5**

**}**

**}**

**{**

**average = ($3 + $4 + $5) / 3**

**if (average < 300) {**

**print $1, "$" average**

**}**

**}**

As shown below.

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Explanation:** The first portion of the script is shebang line **#!/usr/bin/awk -f** which tells how to execute the script and where the instructions are. The **{FS = ":"}** sets the delimiter for the script to a colon the block

**/ Took/ {**

**split($1, name, " ")**

**firstName = name[1]**

**tookMoney = $3 + $4 + $5**

**print firstName, "$" tookMoney**

**}**

Matches the pattern of characters in the slashes **Took** followed by a space then it creates an array with **split($1, name, " ")** created from the first field **$1** called **name** that uses a space **“ “** as a delimiter I make a variable called **firstName** which is the first field of the the **name[]** array **name[1]** I also create a variable called **tookMoney** which adds gets the from the 3 contributions from fields 3 – 5 **$3 $4 $5** I then print **firstName** with a comma for space I put quotes a dollar sign as a string to print the variable **tookMoney**

The next block

**{**

**if ($5 > 10 && $5 < 200){**

**print $1, "$", $5**

**}**

**}**

The **if ($5 > 10 && $5 < 200)** sets a condition that if the last contribution which is in the 5th field **$5** is greater than 10 and **&&** is less than 200 then print the full name which is the first field **$1** I put in quotes a dollar sign to print that as a string

The third block

**{**

**average = ($3 + $4 + $5) / 3**

**if (average < 300) {**

**print $1, "$" average**

**}**

**}**

Creates a variable called **average** which calculates the average of the contributions the line below sets a condition that if the variable **average** is less than 300 then in the segment below wrapped in brackets print the first segment of the code **$1** which is the full name then I put in quotes a dollar sign to print that as a string then the variable average.

To save you would press esc then then type :wq then enter to save and quit to execute the script as it as shown in the picture you need to change permission with the command **chmod u+x** **awkScript.awk** the command **chmod** changed permissions for file the **u** is for users and the **x** is for execution permissions



You may have noticed that the output is not in order and all over the place this is because of how awk processes its scripts. Because of the nature of awk it has been difficult to find a way to get it in order as a compromise I separated each block into separate scripts with the only difference being that the delimiter is set in each script and it uses the **BEGIN** function to have the delimiter set at the start and anything I please at the start so I have a message printed to show what’s happening

A computer screen shot of a black screen

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Sources:**

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

<https://www.tecmint.com/write-shell-scripts-in-awk-programming/>

<https://linuxhint.com/conditional_statement_awk_command/>

<https://stackoverflow.com/questions/19289060/awkhow-to-get-a-part-of-a-field>

<https://stackoverflow.com/questions/48896366/how-to-get-size-of-array-in-awk>

<https://stackoverflow.com/questions/12204192/using-multiple-delimiters-in-awk>

<https://stackoverflow.com/questions/14212993/regular-expression-to-match-a-pattern-inside-awk-command>

<https://unix.stackexchange.com/questions/247548/awk-print-out-file-with-a-row-number>

<https://askubuntu.com/questions/493642/how-to-print-all-columns-using-awk>

<https://stackoverflow.com/questions/65716327/using-awk-to-find-and-replace-strings-in-every-file-in-directory>