AWK

SORTING:-

```
n = asorti(<arraytosort>,<newarray>)
```

The command will get a sorted list of <arraytosort> keywords and put them in <newarray> and n is number of elements in <arraytosort>

And then use

```
for (i = 1; i <= n; i++) {

        key = sorted[i]

        print key " : " sum[key]

    }
```

You can even give your own comparison function to the func!

```
asorti(<arraytosort>,<newarray>,<comparisonfunc>)
```

CRTERIA OF COMPARIOSN FUNCTION

Takes two inputs a and b cmp(a,b)

If it returns < 0 then a will be before b

If it return = 0 then a is considered equal to b

If it returns > 0 then a is after b

Indices are passed to the cmp function remember if you want to use values in array you have to do arr[a] or things like that

OR if you want to do more inefficiently BUT cooler

```
while(1){

        count = 0

                for (key in array){

                        count++

                }

                if(count == 0){
                    break
                }
                keyes = ""
                for (key in array){
                    if(keyes == ""){
                        keyes = key
                    }
                    if(key < keyes){
                        keyes = key
                    }
                }
                print keyes " : " array[keyes]
                delete array[keyes]
        }
```

PRINTING:-

print “hello” “world” this will concatenate and keep “helloworld” while using print “hello”,”world” will separate them with a field separator

Every print statement gives newline at end if you dont want that use printf instead

printf “%s” $1

Arithmetic

If the variable is a string then when you use in numerical equation it acts like 0.

LOOPS

For numeric loop

```
for (i = 1; i <= 10; i++) {
    print i
}
whileloop
while (i <= 5) {
    print i
    i++
}
for (key in arr) {
    print key, arr[key]
}
```

IFELSE

```
{
    if ($1 > 10) {
        print $1 " is greater than 10"
    } else {
        print $1 " is less than or equal to 10"
    }
}
```

regex checking?

string ~ /regex/

Normal regex not extended

. is anything

^ start $ end * 0 or more + 1 or more ? 0 or 1 {} for numbering of amount

\b word boundary

() grouping | pipe is OR

Eg to check if number

```
$1 ~ /^[0-9]+(\.[0-9]+)?$/
```

SED

sed -i changes files in file itself

Without -i it output the changed things in stdout

-n prints only changes lines

sed '10q' bigfile.txt is basically head