stands for Aho, Weinberger, and Kernighan the developers
for manipulating and generating reports
no compiling
command programming language
allows the user to use variables, numeric functions, string functions, and logical operators
tiny but effective programs in the form of statements
it scans a file line by line, splits each input line into fields, compares input line/fields to pattern, performs action(s) on matched lines

syntax:
awk [options] 'selectionCriteria {action}' inputFile > outputFile
where options can be
-v var=val set variable
-F x use x as seperator
-f fileA read program from a fileA

lets have a file sample.txt (report), collection of data in format (name type id). Data seperator in this file is space and record (line) seperator is line break.

```
[root@localhost ~]# echo -e "a stud 123\nb stud 342\nc staff s111\nd staff s234\ne stud 455\nf stud 456" > sample.txt
[root@localhost ~]# ls
dos       hello.c    hello.js    sample.txt
```

now lets print content using awk

```
[root@localhost ~]# awk '{print}' sample.txt
a stud 123
b stud 342
c staff s111
d staff s234
e stud 455
f stud 456
```

now lets print data of only students

```
[root@localhost ~]# awk '/stud/ {print}' sample.txt
a stud 123
b stud 342
e stud 455
f stud 456
```

now lets print only name and id of students
[root@localhost ~]# awk '/stud/ {print $1, $3}' sample.txt
a 123
b 342
e 455
f 456

here $1 holds name $2 holds type $3 holds id in each line, basically they indicates column number. They are built in variables of awk command.
awk reads a file line by line (record) and then breaks line into columns (fields)
$0 indicate entire line
[root@localhost ~]# awk '/stud/ {print "data:", $0}' sample.txt
data: a stud 123
data: b stud 342
data: e stud 455
data: f stud 456

other built in variables are
NR line count (record number)
Nf column count of a line (fields of line)

lets print 'line' lineNumber entireContentOfLine of students from sample.txt
[root@localhost ~]# awk '/stud/ {print "line",NR, $0}' sample.txt
line 1 a stud 123
line 2 b stud 342
line 5 e stud 455
line 6 f stud 456

lets print only last field of staff from sample.txt
[root@localhost ~]# awk '/staff/ {print $NF}' sample.txt
s111
s234

lets print second last column of staff from sample.txt
[root@localhost ~]# awk '/staff/ {print $(NF-1)}' sample.txt
staff
staff

now lets print students from first five lines of sample.txt
[root@localhost ~]# awk 'NR==1, NR==5 {print $0}' sample.txt | awk '/stud/ {print $0}'
a stud 123
b stud 342
e stud 455

is there any empty line?
[root@localhost ~]# awk 'NF==0 {print $0}' sample.txt

variables in awk
lets print 11 to 20 in awk
[root@localhost ~]# awk 'BEGIN { for(i=11;i<=20;i++) print i}'
11
12
13
14
15
16
17
18
19
20

now lets try to create awk script file and execute it, write vi prg.awk and press i to insert mode
[root@localhost ~]# vi prg.awk
#!/usr/bin/awk -f
BEGIN{
for(i=11;i<=15;i++)
print i
}

press ESC to stop inserting and then press :wq for save & quit vi editor
here #! specifies an interpreter for the instructions in a script,
/usr/bin/awk is the interpreter
-f interpreter option, used to read a program file

now lets make script executable and then run it
[root@localhost ~]# chmod +x ./prg.awk
[root@localhost ~]# ./prg.awk
11
12
13
14
15

now lets print details of all students in sample.txt to generate report
in same way as above write following code for prg2.awk
[root@localhost ~]# awk '{print}' sample.txt
a stud 123
b stud 342
c staff s111
d staff s234
e stud 455
f stud 456

[root@localhost ~]# awk '{print}' prg2.awk
#!/usr/bin/awk -f
#read sample.txt and print data of students only
/stud/ {print $0}

[root@localhost ~]# chmod +x prg2.awk
[root@localhost ~]# ./prg2.awk sample.txt
a stud 123
b stud 342
e stud 455
f stud 456

awk script for fibonnaci series
[root@localhost ~]# cat prg3.awk
#!/usr/bin/awk -f
BEGIN
{
a=0;

```
b=1;
N=8;
for(i=0;i<N;i++)
{
print a;
c=a+b;
a=b;
b=c;
}
}
[root@localhost ~]# chmod +x prg3.awk
[root@localhost ~]# ./prg3.awk
0
1
1
2
3
5
8
13
```

now lets read a command line input in awk file and decide if it is even or odd

```
[root@localhost ~]# cat evenOdd.awk
#!/usr/bin/awk -f
BEGIN
{
num = ARGV[1];
if(num%2==0)
        print num, "is even";
else
        print num, "is odd";
}
[root@localhost ~]# chmod +x evenOdd.awk
[root@localhost ~]# ./evenOdd.awk 13
13 is odd
[root@localhost ~]# ./evenOdd.awk 10
10 is even
```