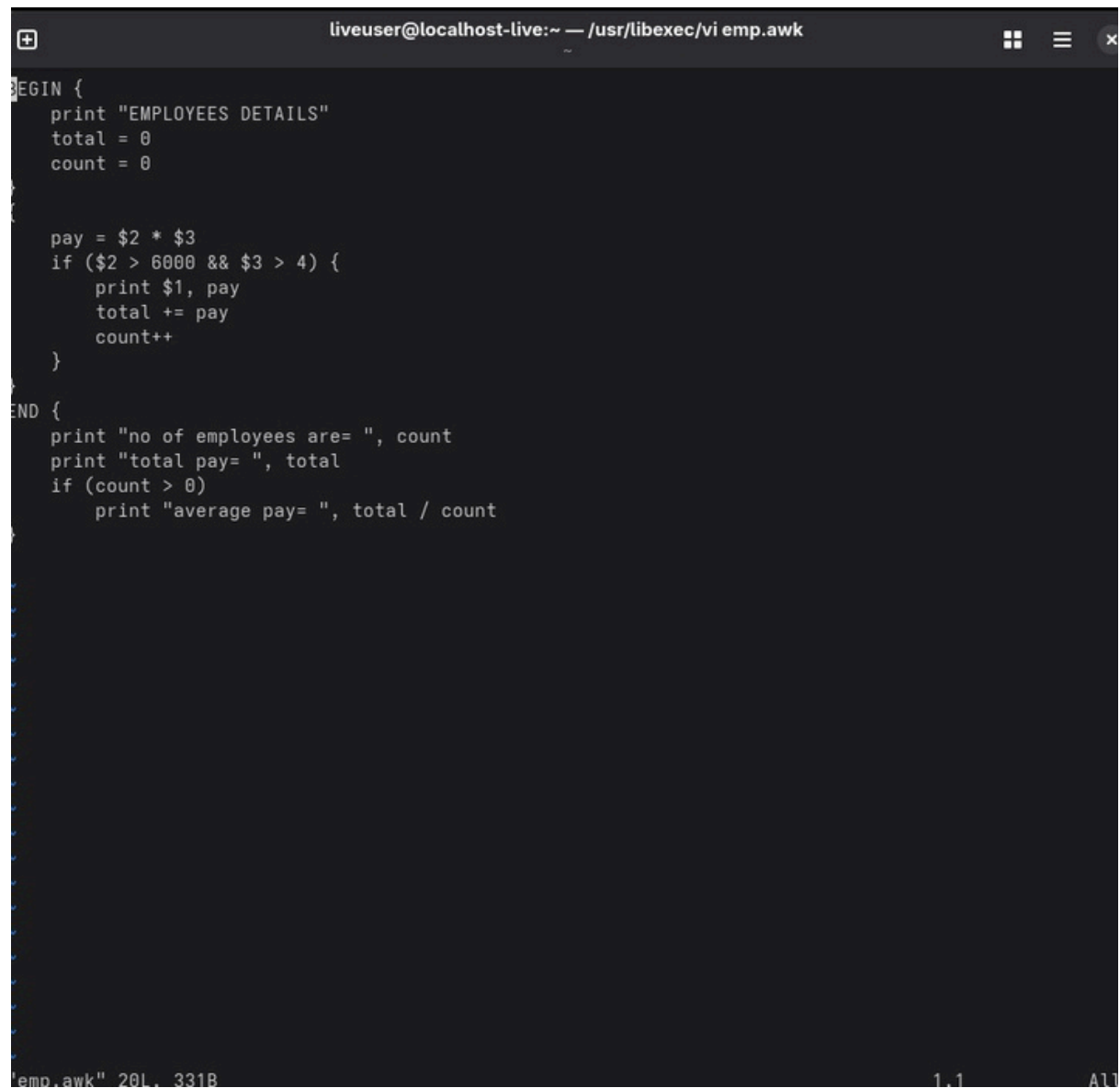


EX 3 AWK

AJAY SRINIVAS R

230701017

EMPLOYEE AVERAGE PAY



The image shows a terminal window with a dark background. The title bar at the top reads "liveuser@localhost-live:~ — /usr/libexec/vi emp.awk". The terminal content displays an AWK script. The script starts with "BEGIN {" followed by "print 'EMPLOYEES DETAILS'", "total = 0", and "count = 0". Then, it calculates "pay = \$2 * \$3" and enters an "if (\$2 > 6000 && \$3 > 4) {" block. Inside this block, it prints "\$1, pay", adds "pay" to "total" ("total += pay"), and increments "count" ("count++"). The script ends with "END {" followed by "print 'no of employees are= ', count", "print 'total pay= ', total", and an "if (count > 0)" block that prints "average pay= ", total / count". The bottom status bar shows "emp.awk" 20L, 331B on the left, "1.1" in the center, and "A11" on the right.

```
BEGIN {
    print "EMPLOYEES DETAILS"
    total = 0
    count = 0

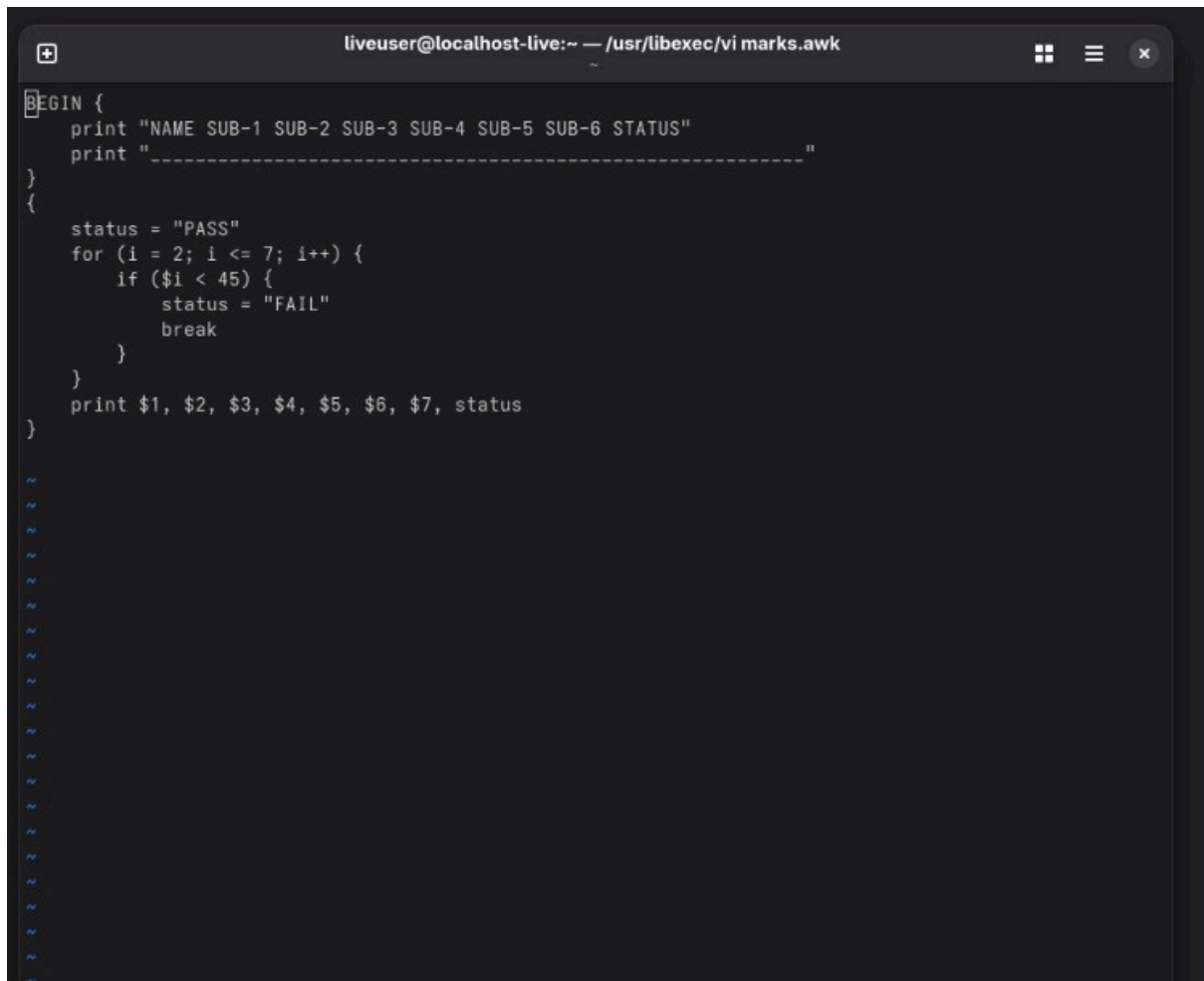
    pay = $2 * $3
    if ($2 > 6000 && $3 > 4) {
        print $1, pay
        total += pay
        count++
    }
}

END {
    print "no of employees are= ", count
    print "total pay= ", total
    if (count > 0)
        print "average pay= ", total / count
}
```

emp.awk" 20L, 331B 1.1 A11

```
liveuser@localhost-live:~  
liveuser@localhost-live:~$ vi emp.dat  
liveuser@localhost-live:~$ vi emp.awk  
liveuser@localhost-live:~$ gawk -f emp.awk emp.dat  
EMPLOYEES DETAILS  
JOE 40000  
BEN 49000  
AMY 39000  
no of employees are= 3  
total pay= 128000  
average pay= 42666.7  
liveuser@localhost-live:~$
```

RESULTS OF EXAMINATION



The image shows a terminal window with a dark background. The title bar at the top reads "liveuser@localhost-live:~ — /usr/libexec/vi marks.awk". The terminal contains an awk script that processes examination data. The script starts with a BEGIN block that prints a header line and a separator line. It then enters a loop that processes each line of input. For each line, it checks if the score in the 7th field is less than 45. If so, it sets the status to "FAIL" and breaks the loop. Otherwise, it sets the status to "PASS". Finally, it prints the name, scores for subjects 1 through 6, and the status.

```
BEGIN {
    print "NAME SUB-1 SUB-2 SUB-3 SUB-4 SUB-5 SUB-6 STATUS"
    print "-----"
}
{
    status = "PASS"
    for (i = 2; i <= 7; i++) {
        if ($i < 45) {
            status = "FAIL"
            break
        }
    }
    print $1, $2, $3, $4, $5, $6, $7, status
}
```

Below the script, there are several lines of tilde (~) characters, likely representing input data or output results.

```
liveuser@localhost-live:~$ vi marks.dat
liveuser@localhost-live:~$ vi marks.awk
liveuser@localhost-live:~$ gawk -f marks.awk marks.dat
NAME SUB-1 SUB-2 SUB-3 SUB-4 SUB-5 SUB-6 STATUS
-----
BEN 40 55 66 77 55 77 FAIL
TOM 60 67 84 92 90 60 PASS
RAM 90 95 84 87 56 70 PASS
JIM 60 70 65 78 90 87 PASS
      FAIL
liveuser@localhost-live:~$
```