```
# aw0010.awk: Provide formatting to data and display the data

# Main processing Loop
{
 printf "\n [%4d] [%-10s] [%7.2f]", $1,$2,$3;
}

# aw0020.awk: BEGIN and END block implementation

BEGIN {
print "\nBegin block processing";
sum=0;
}

# Main processing Loop
{
 printf "\n [%4d] [%-10s] [%5d]", $1,$2,$3;
 sum+=$3
}

END {
print "\n End block processing";
print "\n Total Price : ", sum;
}

# aw0030.awk: Searching for a pattern in a record field
# Look for a Book name starting with either A or C

# Main Processing Loop
{
  if($2 ~ /^[ac]/) { print "Book found ! : ", $0; }
}



# aw0040.awk: if stmt

# Main processing Loop
{
  if ($2 == "acad" && $1 == 1) { printf "\n acad book found: [%21s]", $0;
}
}

# aw0050.awk: While loop

BEGIN {
cntr=0;
while (cntr < 9) {
if (cntr == 0 ) { cntr++ ; continue; }
if (cntr == 4)  { print "counter is 4. Ending loop."; break; }

print "\n cntr : " , cntr++; }
}
```

```
# Main processing Loop
{
}

# aw0060.awk : Print all records falling within a certain range

BEGIN {
      counter=0; ul=0; ll=0;
      print "Lower limit:" ; getline ll < "/dev/stdin";
      print "Upper limit:" ; getline ul < "/dev/stdin";
       }

# Main processing
{
 if ($3 <= ul && $3 >= ll) {
 counter++;
 print "Record found:",$0;
}
}
END {
     printf "\n\n Total records found : %d\n\n", counter ;
    }

# aw0070.awk: NR and FNR

BEGIN {
print "NR : ", NR, "FNR : ", FNR ;
}

# Main processing Loop
{
 print $0;
}

END {
print "END block. NR : " , NR , " FNR : " , FNR;
}


# aw0080.awk: Demo of  using STDIN as input file

# Main processing Loop
{
 printf "\n [%4d] [%-10s] [%7.2f]", $1,$2,$3;
}

# aw0090.awk: Computed fields
BEGIN {
}
# main processing
# increase book price and display each record in proper format
{

  NewPrice = $3 + 10;
```

```
  printf "\n Book : [%10s]    Price : [%7.2f]",$1,NewPrice;

} # main processing block
END { printf "\n"; }


#  Name         : aw0100.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : For loop
#
BEGIN { i=1; }
# record level processing
{
   printf "\n";
   printf("\n Record: %d NF: %d \t Record: [%s] ",NR,NF,$0);
   for (i=1; i<=NF; i++)
   {
      printf "[%s] ",$(i);
   } /* for */
} /* record level processing */

#  Name         : aw0110.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : next keyword
#

# Main processing Loop
{
# skip third record. For allrecords, display last field data
  printf "\n Record Number : %d", NR;
  if (NR == 3 ) { next; }  else { printf "\n%s", $(NF); }
  print "Done";
}

#  Name         : aw0120.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Basic array program
#

BEGIN {
      i=1; arr[1]="one"; arr[2]="two";
      while(i<3) {print arr[i++];}
}

# Main processing Loop
{
}



#  Name         : aw0130.awk
```

```
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Arrays: using IN keyword
#
BEGIN {

a[1]=10; a[3]=30;
i=0;
for(i=0;i<5;i++) {
if(i in a)
{
  print "Arr index " i " has value: " a[i];
}
else { print "Arr index " i " not found"; }
} # if
} # BEGIN
# Main processing loop
{ }
#  Name         : aw0140.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : For and Arrays IN combination: works like FOREACH
#
BEGIN {
a[1]=1; a[3]="three"; a[4]="four";
indx=0;
for(indx in a)
{
  print "Arr index " indx " has value: " a[indx];
}
}

#  Name         : aw0150.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Redirecting output of printf
#
BEGIN  {
      OutFile="";
      print "Specify name of output file: "; getline OutFile <
"/dev/stdin";
      }

# Main processing loop
{
  printf "\n[%6d],[%10s],[%7d]", $1,$2,$3 >> OutFile;
}
#  Name         : aw0160.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  :
#

BEGIN { }
```

```
# main processing
{
        if ($3 <= 200)
        {
                printf "\n%s", $0 >> "/dev/stderr";
        }
        else
        {
                printf "\n[%s]",$0 >> "/dev/stdout";
        } #if
} # main processing

#  Name         : aw0170.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : ARGC and ARGV
#
BEGIN {
print "Argument count:" ARGC;
for(i=0;i<=ARGC;i++)
{
  print ARGV[i];
}
}
# main processing
{  }

#  Name         : aw0180.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Script portability: passing data at run time
#
BEGIN {
         FS=",";
      }
# Main prcessing loop
{
  if(file_code==1)  { printf ("\n Code: %6d Name: %10s Price: %7.2f",
$1,$2,$3);       }
  if(file_code==2)  { printf ("\n Code: [%6d] Name: [%10s] Price:
[%7.2f]", $1,$2,$3); }
}

#  Name         : aw0190.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : OFS demo
#  If OFS is used, it is necessary to re-build the record in memory
#  for this we do $1=$1 which apparently looks like a absurd expression.

BEGIN {
OFS=":"; }
# main processing
{
```

```
 if ($1 > 3) {
     OFS=";";
     $1=$1;  # re-build the record
     print $0; }
  else { $1=$1; print $0; }
}

#  Name         : aw0200.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Record seperator. RS=" " . seperate records by space.
#
BEGIN {
RS=" " ;}

# Main processing Loop
{
  print "Record -> " $0;
}

#  Name         : aw0210.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Will be called form shell script. Parameter driven.
#

# main processing loop
{
# the fld value will be supplied by parent script.

if (fld > NF) { print "Invalid field number specified"; } else {
  print "Field data: " $(fld);}

}
#  Name         : aw0220.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : getline: simple use to read 1 record in addition to
nomral awk cycle
#

BEGIN { print "HEADER";
# the first getline will force open the file and read a record from it
breaking the awk cycle
getline; print $0;
getline; print $0;
print "NR : " NR; # will print the record number
} # begin
# main processing
{print "Main processing cycle block"; print $0; } #main processing
END { print "END" ; }

#  Name         : aw0230.awk
#  Author       : Sameer Ratnaparkhi
```

```
#  Date Written : 29 MAR 2016
#  Description  : getline: Store getline record to a variable
#

BEGIN { print "Begin block";


# the first getline will force open the file and read a record from it
breaking the awk cycle

srcchar="";  # character to be searched in the record
currec=""; # current record

print "Specify search character:"; getline srcchar < "/dev/stdin";
strcmd="cat bmast | grep '"srcchar"'";
print strcmd;

while("1") {
retval = strcmd | getline currec;
if (retval > 0)  { print currec;}
if (retval == 0) { print "Done printing all records. NR:" NR; break; }
if (retval < 0)  { print "Exception caught performing getline..."; break;
}
}
close(strcmd);

} # begin
# main processing
{
  print "Main processing loop";
}
END { print "END" ; }

#  Name         : aw0240.awk
#  Author       : Sameer Ratnaparkhi
#  Date Written : 29 MAR 2016
#  Description  : Accept file name from user and display contents
#

BEGIN {
flname=""; # file to be opened
currec=""; # currently read record

print "Enter Filename to read from: " ;
getline flname < "/dev/stdin";
input_file=flname;
print "file specified:" input_file;

while("1") {
retval = getline currec < input_file;
if (retval == 0) { print "End of file reached"; break; }
if (retval <  0) { print "Exception accessing the file. Exiting..";
break; }
if (retval >  0) { print "Record: " currec; }
```

```
}

}
# Main processing
{}
```