BUILT-IN FUNCTIONS

Lab: SPLIT() function

```
Split function breaks the string into pieces separated by the specific symbol
and stores them into an array
```

```
BEGIN {
str = "abc#xyz#mno#pqr#123";
split (str,s1,"#");
x=1;
for (x in s1) { print s1[x++]; }
} # BEGIN

#MAIN CODE
{}
```

Lab: Length built in function

```
BEGIN { ln=0; }
# record level processing
{
  ln=length($0);
  printf "\nNR:%d",NR;
  printf "\n%4d,%10s,%7.2f. Lenght=%4d",$1,$2,$3,ln;
} /* record level processing */
```

Lab: String concatenation - no built in function required!

```
BEGIN {
s1="abc"; s2="xyz"; s3="";
s3 = "xyz" s1 "***" ;
print s3;
}
# main block
{}

NOTE:
OUTPUT:
$ awk -f str.awk
xyzabc***
```

Lab: Determine how many times $1 value repeated in given file

```
BEGIN {
i=1;LastData=1;}
{
arr[$1]++;            ← increment count by one index $1.default value wull be zero. will become
1.
print arr[$1]; }      ← print the count of number of occurrences of $1
END {
 for (str  in arr) {  ← parse all elements. each element stored in str
 printf "\nstr: %s  arr:%s",str,arr[str];
}
```

Lab: Generate random number

```
BEGIN {
a = 12.6; b=0;
b = int(a); print "int of 12.6 is: ", b;
b=rand(); print "Random number:",b;
} # BEGIN

# main processing
{}
$ awk -f bif2.awk
```

```
int of 12.6 is:  12
Random number: 0.487477
```

NOTE: **PROBLEM WITH RAND() FUNCTION: REPEATS THE SAME VALUE SERIES FOR EVERY INVOCATION**

Lab: use SRAND to specify seed value for generating random numbers

```
BEGIN {
b=0;
srand(4); # 4 is the seed. Different series will be generated if seed is different
b=rand(); print "Random number:",b;
} # BEGIN
# main procesing
{}
```

Lab: Let system generate the seed value for gerating truly random numbers

```
BEGIN {
b=0;
srand(); # system date and time will be used to create seed.
b=rand(); print "Random number:",b;
} # BEGIN
# main procesing
{}
```

NOTE: Since the system date and time is being used, a truly random number is generated each time the program is called

Lab: sprint() function

```
BEGIN {s = ""; }
{
s = sprintf("CODE = [%4d] NAME = [%10s]",$1,$2);
print s;
}
```

Lab: Index() function

```
BEGIN {
str = "abcdefg"; src="c"; pos=0;
pos = index(str,src);
print pos;
}
# main processing code
{}
```

NOTE: INDEX: searches for first occurrence of src in str (base string).

Lab: Match() function

```
BEGIN {
str="001 Apple 120"; regex = "A[pqr]..."; pos=0;
pos=match(str,regex); print pos;
}
# main processing
{}
```

NOTE:
match() locates data which matches with the regular expression and returns the position of this data in the string

Lab: match() function : more uses

```
BEGIN {
# study of match function

reg="a.+"; str="xx apppple xxx appple xx apple ";
retval=match(str,reg);
ln = length(str);
print "retval = " retval;
print "RSTART = " RSTART;
print "RLENGTH = " RLENGTH;
print "length of str: " ln;
}
# main processing code
{ }
```

NOTE:
OUTPUT

```
$ awk -f mat -
retval = 4                    ← match found at 4th position. Thus 4 returned.
RSTART = 4                    ← match found at 4th position
RLENGTH = 28                  ← length of remaining string (from point where match found)
length of str: 31             ← length of entire string
```

NOTE: In AWK, strings being with index 1. If no match found, a 0 is returned.

Lab: Sub() function (substitute)

```
BEGIN {
str="daacaaxaamaa"; repl="T";
s = sub(/aa/,repl,str);
print s;
print str;
}
# main processing block
{}
```

NOTE: SUB replaces contents of existing string with replacement after matching with a regular expression - only the first occurrence. Thus, first occurrence of "aa" got replaced by "T"

Lab: Practical use of sub() function

Replace a PIPE with a "AMPERSAND" symbol

```
BEGIN {
str="daa|caa|xaa|maa"; repl="\\&";
s = sub(/\|/,repl,str);
print s;
print str;
}
# main processing block
{}
```

Lab: gsub() function

gsub: Global substitution

```
BEGIN {
 str = "daa_caa_xaa_maa"; repl="#";
 s   =  gsub(/_/,repl,str);
 print "Output: " str;
 print "Number of substitutions: " s;
}
# main processing block
{}
```

Lab: Tolower() and Toupper() functions

```
BEGIN {
str="ABCDEFG";
str1=tolower(str);
print str1;
str2=toupper(str1);
print str2;
}
# main processing block
{}
```

Lab: substr() function

```
BEGIN {
str="ABCDEFG";
start=2;
num_of_chars=4;

str1=substr(str, start, num_of_chars);

print "Base string:" str;
print "Start position and number of characters:" start, num_of_chars;
print "Output:" str1;
}
# main processing block
{}

NOTE:
OUTPUT:
$ awk -f bif11.awk fr
Base string:ABCDEFG
Start position and number of characters: 2 4
Output:BCDE
```

Lab: system() function

```
BEGIN {
retcode=0;
retcode=system("cp bif12.awk bif_delme.awk");
print "system BIF completed execution. Return code:" retcode;
}
# main processing block
{}
```

NOTE:

1. **Return value:0 : success. <nonzero value>: fail.**
2. **Output of this command executed by system cannot be accessed in AWK script.**
   a. **if you need to access o/p : use cmd1 | getline**