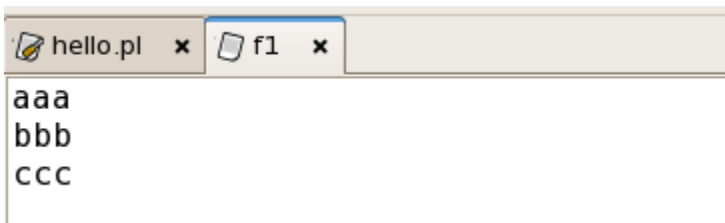


1) Write a program that acts like *cat*, but reverses the order of the lines of all the lines from all the files specified on the command line or all the lines from standard input if no files are specified.

```
#!/usr/bin/perl
print reverse <>;

,---(13:14:39)--[/root]
+-root@localhost> ./hello.pl
omnia
hana
rana
rana
hana
omnia
```

2) Write a program to read in a filename from `STDIN`, then open that file and display its contents with each line preceded by the filename and a colon. For example, if `f1` was read in, and the file `f1` consisted of the three lines `aaa`, `bbb`, and `ccc`, you would see `f1: aaa`, `f1: bbb`, and `f1: ccc`.



```
#!/usr/bin/perl
print "please enter filename: ";
chomp($name = <STDIN>);
open(IN, "$name") || die "cannot open $name: $!";
while (<IN>) {
    print "$name: $_";
}
```

```
,---(13:24:53)--[/root]
+-root@localhost> ./hello.pl
please enter filename: f1
f1: aaa
f1: bbb
f1: ccc
```

3) Write a program to read in a list of filenames and then display which of the files are readable, writable, and/or executable, and which ones don't exist.

```
#!/usr/bin/perl
while (<>) {
    chomp;
    print "$_ is readable\n" if -r;
    print "$_ is writable\n" if -w;
    print "$_ is executable\n" if -x;
    print "$_ doesn't exist\n" unless -e;
}
```

```
,---(14:01:38)--[/root]
+-root@localhost> ./hello.pl
hello.pl
hello.pl is readable
hello.pl is writable
hello.pl is executable
fi
fi doesn't exist
file1.txt
file1.txt is readable
file1.txt is writable
f1
f1 is readable
f1 is writable
```

4) Write a program to change directory to a location specified as input, then list the names of the files in alphabetical order after changing there. (Don't show a list if the directory change did not succeed: merely warn the user.)

```
#!/usr/bin/perl -w
print "enter the directory: ";
chomp($dir = <STDIN>);
chdir($dir) || die "Cannot chdir to $dir $!";
print "the names of the files in alphabetical order: \n";
foreach $file (sort glob("*")) {
    print "$file\n";
}
```

```
,---(14:21:53)--[/root]
+-root@localhost> ./hello.pl
enter the directory: dirrrrrr
Cannot chdir to dirrrrrr No such file or directory at ./hello.pl line 4, <STDIN> line 1.

,---(14:21:59)--[/root]
+-root@localhost> ./hello.pl
enter the directory: dir1
the names of the files in alphabetical order:
dir12
file5.txt
lab
lab.txt
```

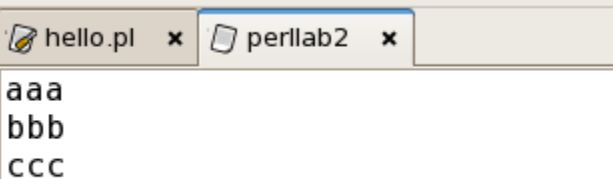
5) Write a program that works like *rm*, deleting the files given as command-line arguments when the program is invoked.

```
#!/usr/bin/perl -w
unlink (@ARGV) || die "failed to delete files:$!\n"
```

6) Write a program that works like *mv*, renaming the first command-line argument to the second command-line argument.

```
#!/usr/bin/perl -w
($old_name, $new_name) = @ARGV;
rename($old_name, $new_name) || die "Cannot rename $old_name to $new_name: $!";
```

```
,---(15:32:44)--[/root]
+-root@localhost> ./hello.pl f1 perllab2
```



```
aaa
bbb
ccc
```

7) Write a program that works like *ln*, creating a hard link from the first command-line argument to the second.

```
#!/usr/bin/perl -w
($name1, $name2) = @ARGV;
link($name1, $name2) || die "Cannot create hard link from $name1 to $name2: $!";
```

```
,---(15:57:56)--[/root]
+-root@localhost> ./hello.pl file1 lab_link
```

8) Modify the program from the previous exercise to handle an optional `-s` switch to create soft link.

```
#!/usr/bin/perl -w
$sym=0;
if ($ARGV[0] eq '-s') {
shift @ARGV;
$sym= 1;}
($name1, $name2) = @ARGV;
if($sym==1){
symlink($name1,$name2)|| die "Cannot create symbolic link from $name1 to $name2: $!";
}else{
link($name1,$name2)|| die "Cannot create hard link from $name1 to $name2: $!";
}
```

```
,---(16:07:57)--[/root]
+-root@localhost> ./hello.pl -s file1 lab_slink
```

9) Write a program that looks for all soft link files in the current directory and prints out their name and their original file similar to the way `ls -l` does it (name -> value).

Create some soft links in the current directory and test it out.

```
#!/usr/bin/perl -w
@files = glob("*");
foreach $file (@files) {
$f = readlink $file;
if (defined $f) {
printf("%s -> %s\n", $file, $f|);}
}
```

```
,---(16:18:31)--[/root]
+-root@localhost> ./hello.pl
lab_slink -> file1
lab_slink1 -> file2.txt
---(16:19:01)--[/root]
```

10) Write a subroutine to take a numeric value from 1 to 9 as an argument and return the English name (such as one, two, or nine). If the value is out of range, return the original number as the name instead.

```
#!/usr/bin/perl -w
sub num_names {
    my(%names);
    my @names= qw(one two three four five six seven eight nine);
    my($num) = @_;
    return ($num >= 1 && $num <= 9) ? $names[$num - 1] : $num;
}
while (<>) {
    chomp;
    print "number $_ is ", num_names($_), "\n";
}
```

```
,---(15:14:44)--[/root]
+-root@localhost> ./hello.pl
1
number 1 is one
2
number 2 is two
3
number 3 is three
5
number 5 is five
6
number 6 is six
8
number 8 is eight
9
number 9 is nine
10
number 10 is 10
11
number 11 is 11
```

11) Taking the subroutine from the previous exercise, write a program to take two numbers and add them together, displaying the result as Two plus two equals four. (Don't forget to capitalize the initial word!).

```
#!/usr/bin/perl -w
sub num_names {
    my(%names);
    my @names= qw(one two three four five six seven eight nine);
    my($num) = @_ ;
    return ($num >= 1 && $num <= 9) ? $names[$num - 1] : $num;
}
print "enter num1: ";
chomp($num1= <STDIN>);
print "enter num2: ";
chomp($num2= <STDIN>);
$add=$num1+$num2;
print ucfirst(num_names($num1)) . " plus " . num_names($num2) . " equals " . num_names($add) . ".\n";

,---(15:31:27)--[/root]
+-root@localhost> ./hello.pl
enter num1: 2
enter num2: 4
Two plus four equals six.

,---(15:32:22)--[/root]
+-root@localhost> ./hello.pl
enter num1: 5
enter num2: 5
Five plus five equals 10.
```