Programming in Perl
Lecture 4 Examples

File modes:

```
Mode Symbol       Meaning
-----------       ---------------------
(nothing)         Open for reading
<                 Open for reading
>                 Open for writing
                    (destroys current contents)
>>                Open for appending
                    (preserves current contents;
                     ready to write at end)
+<                Open for both reading & writing
                    (preserves current contents)
+>                Open for both reading & writing
                    (destroys current contents)
+>>               Open for both reading & writing
                    (preserves current contents;
                     ready to write at end)
```

Die examples:

```
$tempfile = "temp.out";
open(TEMP, ">$tempfile") or die("Cannot open $tempfile\n");

o $! is the error string from the most recent operating system
error

$errorfile = "i.do.not.exist";
open(IN, "$errorfile") or die("Cannot open $errorfile : $!\n");
# Cannot open i.do.not.exist : No such file or directory

o if the error message to be printed does not include a
newline, Perl will automatically give the script name and line
number from which die was called

$errorfile = "i.do.not.exist";
open(IN, "$errorfile") or die("Cannot open $errorfile : $!;");
# Cannot open i.do.not.exist : No such file or directory; at
./die.plx line 80.
```

Upcasing a file (output to second file):

```
print("This script stores the contents of file1 in file2\n");
print("with all letters made uppercase.\n");
print("Enter file1 (the input file): ");
chomp($file = <STDIN>);
print("Enter file2 (the output file): ");
chomp($upper_file = <STDIN>);

open(INFILE, "<$file") or die("Cannot open $file for input : $!\n");
open(UPFILE, ">$upper_file")
  or die("Cannot open $upper_file for output : $!\n");

while(<INFILE>) {
    print UPFILE ("\U$_");
}

close(INFILE) or die("Cannot close $file : $!\n");
close(UPFILE) or die("Cannot close $upper_file : $!\n");
```

Upcasing a file (output to the same file):

```
print("This script changes the contents of the file entered\n");
print("making all letters uppercase.\n");
print("Enter a file name to be upcased: ");
chomp($file = <STDIN>);

# can't use +> because contents would be erased before we did
# anything with them
open(INOUTFILE, "+<$file")
  or die("Cannot open $file for modification: $!\n");

@in_array = <INOUTFILE>;

for($i=0; $i < @in_array; $i++) {
    $in_array[$i] = "\U$in_array[$i]";
}


# seek(FILEHANDLE, OFFSET, WHENCE)
# (OFFSET in bytes; WHENCE=0 means relative to beginning of file)
# positions file pointer back at start of INOUTFILE
seek(INOUTFILE, 0, 0);

print INOUTFILE (@in_array);

close(INOUTFILE) or die("Cannot close $file : $!\n");
```

Function example (return values):

```
sub max_of_x_and_y {
  if($x > $y) {
    return $x;
  } else {
    return $y;
  }
}
```

Function example (arguments):

```
sub say {
  print("$_[0], $_[1]!\n");
}

say("hello", "world");
```

Function example (more arguments and pass by reference):

```
 sub largest {
  if (@_ == 0) {
    print("Warning no parameters passed to largest\n");
    return undef;
  }
  $max = $_[0];
  for($i = 1; $i < @_; $i++) {
    if($_[$i] > $max) {
      $max = $_[$i];
    }
  }
  return $max;
}

sub pass_by_ref {
 for($i = 0; $i < @_; $i++) {
   $_[$i] = "hi";
 }
 return $max;
}

$a = largest(10,200,300,49);
print("$a\n");
@g = (10,200,300,49);
$a = largest(@g);
print("$a\n");

@g = (10,200,3,49);
$a = pass_by_ref(@g);
print("$a\n"); # 300 (global !)
print("@g\n"); # hi hi hi hi
$a = pass_by_ref(10,200,300,49);
print("$a\n");
# Modification of a read-only value attempted at ./largest.plx line 39
# nothing printed
```

MY operator (private variable – lexically scoped):

```
sub largest2 {
 # local variable to allow call by value
 my (@values) = @_;
 # local variable; any assignment will
 # not effect a global $max
 my ($max);
 if (@values == 0) {
   print("Warning no parameters passed to largest\n");
   return undef;
 }
 $max = $values[0];
 for($i = 1; $i < @values; $i++) {
   if($values[$i] > $max) {
     $max = $values[$i];
   }
 }
 return $max;
}
```

LOCAL function (semiprivate variable – dynamically scoped):

```
$value = "original";

tellme();
spoof();
tellme();

sub spoof {
  local ($value) = "temporary";
  tellme();
}

sub tellme {
  print("Value is $value.\n");
}

# prints
Value is original.
Value is temporary.
Value is orginal.
```