

COMP 322 Internet Systems Fall 2006 Assignment 6

Due in the digital drop box by Wednesday, Nov. 8 at 11:00 PM

1. Write a Perl program that does the following. It first prompts for and reads the user's name (first and last names). Then it prompts for and reads the user's middle initial (just the letter, not the period). Finally, it outputs the user's full name (first name, middle initial with a period, and last name) in a 30- or 29-column format. If the sum of the lengths of the first name, the middle initial (with the period), and the last name is an even number, then a 30-column format is used; otherwise, a 29-column format is used. The same number of spaces occurs between the first name and the middle initial as between the middle initial and the last name to bring the width up to 29 or 30 characters. (We assume that no one has a first name and a last name whose lengths sum to more than 26.)

The following is an example run.

```
C:\SomeFolder>perl prob3.pl
Your name: Albert Esterline
Your middle initial: C
Albert  C.  Esterline
```

Here there are six spaces between "Albert" and "C." and also six spaces between "C." and "Esterline".

We give you a statement-by-statement description of the program below. The following variables are used:

- `$name`: The name (first and last) input, with the input record separator chomped off.
- `$space_pos`: The index of the space (between the first and last names) in `$name`
- `$first`: The first name
- `$last`: The last name
- `$MI`: The middle initial
- `$len_used`: The sum of the lengths of the first name, middle initial (with the period), and last name
- `$num_spaces`: The number of spaces to fill in both between the first name and middle initial and between the middle initial and last name. (This may have a fractional part of .5—the repetition operator rounds down.)

The following is the statement-by-statement description:

1. Output the prompt for the name.
2. Assign the input line to `$name` and remove the input record separator.
3. Set `$space_pos` to the index of the space in the name.
4. Set `$first` to the first name.
5. Set `$last` to the last name.
6. Output the prompt for the middle initial.
7. Assign the input line to `$MI` and remove the input record separator.
8. Concatenate a period (.) to the end of the middle initial in `$MI`.
9. Set `$len_used` to the sum of the lengths of the first name, middle initial, and last name
10. Set `$num_spaces` to $(30 - \$len_used) / 2.11$. Output the required string.

This can be done somewhat easier with operators we'll see when we come to arrays. Do not use array operators for this problem.

2. The following Perl code (with fragments missing) is in a file `prob2.pl`. When executed with a command-line argument specifying a file that has one value per line, it first inputs each value and stores it in the next cell in array `@ar2`. It shifts one value at a time from `@ar2` and pushes in onto array `@ar1` until `@ar2` is empty. Before any value is moved and just after each value is moved from `@ar2` to `@ar1`, both arrays are output.

Supply the missing code. You can download this file (with the gaps) from the location where you got this assignment. A data file (to specify on the command line), `prob2.dat`, can also be downloaded.

```
# Input the second array

while ( α ) {
    chomp( $ar2[ $i++ ] = $_ );
}

print "β";

# On each iteration, shift a value from the
# second to the first array

while ( γ ) {
    δ;
    print "β";
}
```

α : Input a value from the file specified on the command line.

β (occurs twice): Output both arrays. Don't use a loop.

γ : An expression that is true if and only if `@ar2` is not empty

δ : Shift a value out of array `@ar2` and push it onto array `@ar1`.

The following are the contents of the data file, `prob2.dat`:

```
1
2
3
```

The following is a run:

```
C:\someDirectory>perl prob2.pl prob2.dat
@ar1:      @ar2: 1 2 3
@ar1: 1    @ar2: 2 3
@ar1: 1 2  @ar2: 3
@ar1: 1 2 3 @ar2:
```

3. The following Perl program (with fragments missing) is in the file prob3.pl. It is self-contained since the hash `%points` is assigned the data, which indicate the points earned by various students. It first uses a `foreach` loop to find the sum of the points for all the students. This sum is in the variable `$total`, and this is output. It then calculates and outputs the average number of points per student (the total divided by the number of students, which is the same as the number of point values). Finally, it finds the largest number of points earned and the name of a student who earned this many points, and it outputs these. (There may be more than one student with the largest number of points; it finds just one.)

Supply the missing code in the following. Each gap is labeled with a Greek letter, which is repeated after the listing with a description of what should fill the corresponding gap. You can download this file (with the gaps) from the item just below the location where you got this assignment.

```

%points = ( "Ed" => 15, "Sue" => 13, "Al" => "17", "Ken" => 14 );

# Sum all the values, giving the value for $total.
foreach α_____
    _____
    -
print "The total points: $total \n";

# Calculate the average points per student.
$ave = β_____ ;
print "The average points per student: $ave \n";

# Find the largest number of points earned (in $max_pts) and
# a student who earned this number of points (in $max_name).
while ( ( $name, $pts ) = γ_____ ) {
    δ_____
    _____
    -
}

print "$max_name has the most points, $max_pts \n";

```

α : Complete the `foreach` loop so that, when it terminates, the variable `$total` has the sum of all the values in hash `%points`. **Hint:** Use the values hash operator.

β : Calculate the average points per student. **Hint:** The operand positions in a divide expression (using the `/` arithmetic operator) are scalar contexts.

γ : Return the next key-value pair in `%points`.

δ : Fill in the body of this loop so that, when it terminates, `$max_pts` contains the largest number of points any of the students received, and `$max_name` contains the name of a student with this number of points. (There may be several students with the largest number of points; you need only identify one.) **Hint:** `$max_pts` is by

default initially 0. As you go through the pairs, `$max_pts` is the largest number of points found so far. If `$pts` in the next pair is greater than `$max_pts`, update `$max_pts` and `$max_name` accordingly.

The following is a run of this program.

```
C:\SomeFolder\HW4>perl prob3.pl
The total points: 59
The average points per student: 14.75
Al has the most points, 17
```