```
01/24/2001 Ahmed's Camel Emporium                  1147.99
01/28/2001 Flea spray                                24.99
01/29/2001 Camel rides to tourists     1235.00
03/23/2001Totals                     1235.001172.98
```

OK, it's a start, but what happened to the spaces? We put `x`, didn't we? Shouldn't it skip forward? Let's look at what `pack` in *perlfunc* says:

```
x   A null byte.
```

Urgh. No wonder. There's a big difference between "a null byte", character zero, and "a space", character 32. Perl's put something between the date and the description - but unfortunately, we can't see it!

What we actually need to do is expand the width of the fields. The `A` format pads any non-existent characters with spaces, so we can use the additional spaces to line up our fields, like this:

```
print pack("A11 A28 A8 A*", $date, "Totals", $tot_income, $tot_expend);
```

(Note that you can put spaces in the template to make it more readable, but they don't translate to spaces in the output.) Here's what we got this time:

```
01/24/2001 Ahmed's Camel Emporium                  1147.99
01/28/2001 Flea spray                                24.99
01/29/2001 Camel rides to tourists     1235.00
03/23/2001 Totals                      1235.00 1172.98
```

That's a bit better, but we still have that last column which needs to be moved further over. There's an easy way to fix this up: unfortunately, we can't get `pack` to right-justify our fields, but we can get `sprintf` to do it:

```
$tot_income = sprintf("%.2f", $tot_income);
$tot_expend = sprintf("%12.2f", $tot_expend);
$date = POSIX::strftime("%m/%d/%Y", localtime);
print pack("A11 A28 A8 A*", $date, "Totals", $tot_income, $tot_expend);
```

This time we get the right answer:

```
01/28/2001 Flea spray                                24.99
01/29/2001 Camel rides to tourists     1235.00
03/23/2001 Totals                      1235.00      1172.98
```

So that's how we consume and produce fixed-width data. Let's recap what we've seen of `pack` and `unpack` so far:

- Use `pack` to go from several pieces of data to one fixed-width version; use `unpack` to turn a fixed-width-format string into several pieces of data.
- The pack format `A` means "any character"; if you're packing and you've run out of things to pack, `pack` will fill the rest up with spaces.
- `x` means "skip a byte" when unpacking; when packing, it means "introduce a null byte" - that's probably not what you mean if you're dealing with plain text.
- You can follow the formats with numbers to say how many characters should be affected by that format: `A12` means "take 12 characters"; `x6` means "skip 6 bytes" or "character 0, 6 times".
- Instead of a number, you can use `*` to mean "consume everything else left".

  Warning: when packing multiple pieces of data, `*` only means "consume all of the current piece of data". That's to say

  ```
  pack("A*A*", $one, $two)
  ```

  packs all of `$one` into the first `A*` and then all of `$two` into the second. This is a general principle: each format character corresponds to one piece of data to be packed.