In coding, you have many dimensions in which you can rate code:

- Brevity of code
- Featurefulness
- Speed of execution
- Time spent coding
- Robustness
- Flexibility

Now, remember, these dimensions are all in opposition to one another. You can spend three days writing a routine which is really beautiful and fast, so you’ve gotten two of your dimensions up, but you’ve spent three days, so the “time spent coding” dimension is way down.

So, when is this worth it? How do we make these decisions? The answer turns out to be very sane, very simple, and also the one nobody, ever, listens to: **Start with brevity. Increase the other dimensions as required by testing.**

I would add clarity, particularity, simplicity and under/over-engineered to the list.

Randall Munroe wrote in [https://xkcd.com/974/](https://xkcd.com/974/):

![XKCD Comic](https://i.xkcd.com/974.png)

---

**Task #1**

From Perl Weekly Challenge - 018, Task #1 retrieved on 2019-07-28 at 23:14 +00:

Write a script that takes 2 or more strings as command line parameters and print the longest common substring. For example, the longest common substring of the strings “ABABC”, “BABCA” and “ABCBA” is string “ABC” of length 3. Other common substrings are “A”, “AB”, “B”, “BA”, “BC” and “C”. Please check this wiki page for details.
Perl 6 solution

```perl
# Perl Weekly Challenge - 018
# Task #1
#
# See
# engineering.purdue.edu/~mark/pwc-018.pdf
# for more information.
#
# Run using Perl v6.d.
use v6.d;

# Make an array of sets.
# Each element of the array will contain a set of all substrings
# for the corresponding word.
my SetHash @set;

# For each word on the command line
for (0 .. @* ARG.S.elems - 1) -> $i {
    # Get the word.
    my $word = @* ARG.S[$i];
    # Compute and store every substring in the word in @set[$i].
    for (0 .. $word.chars - 1) -> $startpos {
        for ($startpos .. $word.chars - 1) -> $endpos {
            my $substr = $word.substr($startpos .. $endpos);
            @set[$i]{$substr} = True;
        }
    }

    # Compute the intersection of all sets,
    # sort by the number of characters,
    # get the last entry (the one with the most characters),
    # and print it.
    ((&)) @set).keys.sort({. chars}).tail.say;
}
```

Task #2

From Perl Weekly Challenge - 018, Task #2 retrieved on 2019-07-28 at 12:16 +00:

Write a script to implement Priority Queue. It is like regular queue except each element has a priority associated with it. In a priority queue, an element with high priority is served before an element with low priority. Please check this wiki page for more informations. It should serve the following operations:

1) is_empty: check whether the queue has no elements.
2) insert_with_priority: add an element to the queue with an associated priority.
3) pull_highest_priority_element: remove the element from the queue that has the highest priority, and return it. If two elements have the same priority, then return element added first.
Perl 6 solution

# Perl Weekly Challenge - 018
# Task #2
#
# See
# engineering.purdue.edu/~mark/pwc-018.pdf
# for more information.

# Run using Perl v6.d.
use v6.d;

# Priorities and values are stored in parallel arrays.
my (@priority, @value);

sub insert_with_priority($priority, $value) {
    @priority.push($priority); @value.push($value);
}

sub pull_highest_priority_element {
    (is_empty) and return Nil;
    my $i = (0..^@priority).sort({@priority[$^a] < @priority[$^b]}).head;
    my $value = @value[$i];
    @priority.splice($i, 1, ()); @value.splice($i, 1, ());
    return $value;
}

sub print_queue {
    "print_queue".say;
    for (0..^@priority) -> $i {
        " { @priority[$i]} { @value[$i]}".say;
    }
}

sub is_empty {
    return !@priority.elems;
}

print_queue;
insert_with_priority(5, "this is the first priority 5 item");
print_queue;
insert_with_priority(5, "this is the second priority 5 item");
print_queue;
insert_with_priority(4, "this is the first priority 4 item");
print_queue;

49  pull_highest_priority_element.say;
50  print_queue;
51  pull_highest_priority_element.say;
52  print_queue;
53  pull_highest_priority_element.say;
54  print_queue;
55  print_queue;