Theory and Practice of Logic Programming **1** (1): 127–128, January 2001.

TPLP pearls submission guidelines

LEE NAISH

Errors, like straws, upon the surface flow;
He who would search for pearls must dive below.
– John Dryden

I am honoured to be the editor of the *Logic Programming Pearls* section of the journal *Theory and Practice of Logic Programming*. Below are some brief guidelines for submission of papers to this section. Note that submissions of pearls in logic programming languages other than Prolog are welcome. Functional programming language pearls, although similar in spirit, are more appropriately submitted to another journal such as the *Journal of Functional Programming*.

A programming pearl is a short piece of self-contained code of outstanding quality. Ideally it should be clearly correct, elegant, concise, efficient, etc., though in some cases a (small) subset of these may not apply. It may be a useful application or may primarily be an example of a useful programming technique. Portability is not so important in this context, but if it can be achieved without sacrificing other qualities, so much the better.

Accompanying text explains the code and its qualities. These may be exposed by describing how a programmer could derive the code. Ideally, a logic programming pearl should also showcase the logic programming paradigm, for example, declarative semantics, nondeterminism, logic variables, definite clause grammars, meta programming, and so forth.

The following classification may help clarify what is considered to be a logic programming pearl.

Natural Pearls Like theorems in mathematics, natural pearls are discovered rather than manufactured. A logic program embodies a formulation of the mathematical logic behind a problem. This logic, plus a logic programming interpreter, induces an algorithm to solve the problem. Things of great beauty can be formed from these elements.

Cultured Pearls Cultured pearls are, in part, artifacts of our technology and are regarded somewhat less highly than natural pearls. They may depend on some particular implementation technique such as the Warren Abstract Machine or a specific language feature.

Black Pearls Black pearls contain impurities but are still beautiful. A program is not rejected as a pearl only on the grounds of impurity.

Cut and Paste Some artifacts may seem like pearls to the untrained eye, but can be distinguished by those of more refined taste. The most ugly scar on the face of Prolog programming is the abused *cut*. Using `goto` in C because you don't understand `while` loops is not a recipe for pearls. Similarly, knowing when to use and not use cut is a prerequisite for good Prolog programming. A good reference for Prolog programming is Richard O'Keefe's *The Craft of Prolog* published by MIT Press.

https://doi.org/10.1017/S1471068400001071 Published online by Cambridge University Press