WLPP Programming Language Tutorial

WLPP in a nutshell for Java or C++ Programmers

A WLPP program is a C++ function named `wain` that has two parameters. The type of the first parameter is either `int` or `int*`. The type of the second parameter is `int`. The type of the result of the `wain` procedure is `int`.

Declarations, control structures and statements that may be used in WLPP are restricted to:

- `int` or `int*` *(declaration of a single int or int* variable with an unsigned integer constant or NULL initializer; all declarations in WLPP must precede all statements and control structures; every declaration must include an integer constant or NULL initializer)*
- `if` *(must have an else clause)*
- `while`
- `return` *(must be the last statement in method)*
- `println`
- `=` *(i.e. assignment)*

The clauses of `if` and `while` containing statements must be enclosed in braces (i.e. {}).

Expressions may contain only variable names, integers (written in decimal without a sign), unary `&` and `*`, and the binary (two operand) versions of the following operators:

```
+ - * / % == != <= >= < >
```

Arrays of consecutive integers may be dynamically allocated using `new` and `delete []`, but their elements can be accessed only using pointer dereferences, because WLPP does not include the C++ operator `[]`. The // notation (and only the // notation) may be used for comments.

Example WLPP Program

```
//
// WLPP Program to compute:
//   a^b if 0 <= a,b < 10
//   -1 otherwise
//

int wain(int a, int b) {
   int counter = 0;
   int product = 0;
   product = 0-1;  // only binary minus
   if (a >= 0) {
      if (b >= 0) {
         if (a < 10) {
            if (b < 10) {
               product = 1;
               counter = 0;
               while (counter < b) {
```

```
                    product = product * a;
                    counter = counter + 1;
                }
            } else {} // must have else
        } else {}
    } else {}
} else {}
return product;
}
```