Homework 3

Declarations take the form of the name of a type followed by a “declarator,” which looks a lot like an expression with just one identifier in it. The meaning is that an evaluation of an expression of that form will yield an object of that type. For example “char **p” says that **p is a char, or in other words, *p is a pointer to char, or in other words, p is a pointer to pointer to char.

The identifier in a declarator can be omitted, producing something that describes a type without declaring any identifier. Thus “char ***” describes the type “pointer to pointer to char.”

Subscript operations ([]) in declarators have no need for particular subscript values, so such values can be omitted; if a value is enclosed within the brackets, it is taken to be the size of the array in question.

Note that the subscript operator [] and the function-call operator () have higher precedence than the dereference operator *. Thus “char *p[]” is equivalent to “char *(p[])” and declares an array of pointers, whereas “char (*p)[]” declares a pointer to an array. Similarly, “char *f()” declares a function named f whose return value is a pointer, while “char (*f)()” declares a pointer named f to a function (whose name is not shown) which returns a char.

In a declarator, unary & acts as a variant version of *; it describes a reference to something instead of a pointer to it. By contrast, in expressions, unary & acts as the inverse of unary *.

Declarators can use the subscript operator ([]), function calls (with the types of the arguments shown explicitly), several flavors of dereference operators, and parentheses for grouping. The dereference operators are as follows. The identifier classname represents the name of a class.

\[
\begin{align*}
&\text{(reference)} \\
\ast &\text{(pointer)} \\
\ast \text{ const} &\text{(constant pointer)} \\
:: \text{classname} \ast &\text{(member pointer)} \\
:: \text{classname} \ast \text{ const} &\text{(constant member pointer)}
\end{align*}
\]

On the following page is a chart. In the first column are C++ declarations for identifier I. In the second column are informal descriptions of the declared types. (Most of the declared types are legitimate and useful; some may not be supported by the compiler and some may not even be logically sensible.) Match them up, drawing a line connecting each declaration to its meaning. Turn the result in at the next class meeting (Wednesday, Feb. 4).
float & * I  
reference to pointer to float

float * & I  
array of reference to pointer to float

float * & I []  
pointer to array of reference to float

float & (*I) []  
pointer to reference to float

float (**I) []  
constant pointer to pointer to float

float *(I)[]  
pointer to constant pointer to float

float (*((I[])))  
pointer to pointer to constant float

float (I)(float *)  
pointer to pointer to array of float

float *I(float)  
pointer to array of pointer to float

float (*I)(float)  
array of pointer to pointer to float

float ** const I  
pointer to function, of float, returning float

float * const * I  
function, of float, returning pointer to float

float const * * I  
function, of pointer to float, returning float