91.10.2 `eval '...'` improvements

Line numbers (as reflected by caller() and most diagnostics) within `eval '...'` were often incorrect where here documents were involved. This has been corrected.

Lexical lookups for variables appearing in `eval '...'` within functions that were themselves called within an `eval '...'` were searching the wrong place for lexicals. The lexical search now correctly ends at the subroutine's block boundary.

The use of `return` within `eval {...}` caused $@ not to be reset correctly when no exception occurred within the eval. This has been fixed.

Parsing of here documents used to be flawed when they appeared as the replacement expression in `eval 's/.../.../e'`. This has been fixed.

91.10.3 All compilation errors are true errors

Some "errors" encountered at compile time were by necessity generated as warnings followed by eventual termination of the program. This enabled more such errors to be reported in a single run, rather than causing a hard stop at the first error that was encountered.

The mechanism for reporting such errors has been reimplemented to queue compile-time errors and report them at the end of the compilation as true errors rather than as warnings. This fixes cases where error messages leaked through in the form of warnings when code was compiled at run time using `eval STRING`, and also allows such errors to be reliably trapped using `eval "..."`.

91.10.4 Implicitly closed filehandles are safer

Sometimes implicitly closed filehandles (as when they are localized, and Perl automatically closes them on exiting the scope) could inadvertently set $? or $!. This has been corrected.

91.10.5 Behavior of list slices is more consistent

When taking a slice of a literal list (as opposed to a slice of an array or hash), Perl used to return an empty list if the result happened to be composed of all undef values.

The new behavior is to produce an empty list if (and only if) the original list was empty. Consider the following example:

```
    @a = (1,undef,undef,2)[2,1,2];
```

The old behavior would have resulted in @a having no elements. The new behavior ensures it has three undefined elements.

Note in particular that the behavior of slices of the following cases remains unchanged:

```
    @a = ()[1,2];
    @a = (getpwent)[7,0];
    @a = (anything_returning_empty_list())[2,1,2];
    @a = @b[2,1,2];
    @a = @c{'a','b','c'};
```

See *perldata*.

91.10.6 `(\$)` prototype and `$foo{a}`

A scalar reference prototype now correctly allows a hash or array element in that slot.