CSC488S/CSC2107S Compilers and Interpreters **Winter 2011/2012**

Mid Term Test Solution

1. [20 marks] The grammar converted to LL(1).

				Predict Set
1	S	$\rightarrow$	c B S1	{ c }
2		$\rightarrow$	B S1	{ a , b , d }
3	S1	$\rightarrow$	c A S1	{ c }
4		$\rightarrow$		{ $ }
5	A	$\rightarrow$	b A	{ b }
6		$\rightarrow$	d	{ d }
7	B	$\rightarrow$	a A B1	{ a }
8		$\rightarrow$	A B2	{ b , d }
9	B1	$\rightarrow$	a A	{ a }
10		$\rightarrow$		{ c , $ }
11	B2	$\rightarrow$	b A	{ b }
12		$\rightarrow$		{ c , $ }

2. [25 marks] a) Yes, something like:

```
 1  program             ::=  majorScope ;
 2  majorScope          ::=  L_CURLEY majorDeclarations statements R_CURLEY
 3                        |  L_CURLEY statements R_CURLEY
 4                        |  L_CURLEY R_CURLEY
 5                      ;
 6  majorDeclarations   ::=  majorDeclaration
 7                        |  majorDeclarations majorDeclaration
 8                      ;
 9  majorDeclaration    ::=  declaration
10                        |  pfDeclaration
11                      ;

30  declaration         ::=  VAR variablenames COLON type
31                      ;
32  pfDeclaration       ::=  FUNCTION IDENT COLON type majorScope
33                        |  FUNCTION IDENT L_PAREN parameters R_PAREN COLON type majorScope
34                        |  PROCEDURE IDENT majorScope
35                        |  PROCEDURE IDENT L_PAREN parameters R_PAREN majorScope
36                      ;
```

It was not necessary to provide a full grammar solution, a clearly written description of a solution like the one above was sufficient.

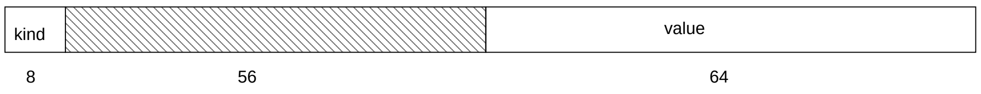
b) Yes. As a minimum, keep a state variable *inMinorScopes* that is incremented when a minor scope is entered in *statement* and decremented when the scope is exited. When a function/procedure declaration is encountered, check that inMinorScopes is zero.

3. [20 marks]

There are a lot of *token separation issues* in Fortran. Some of the issues that a solution could have discussed include:

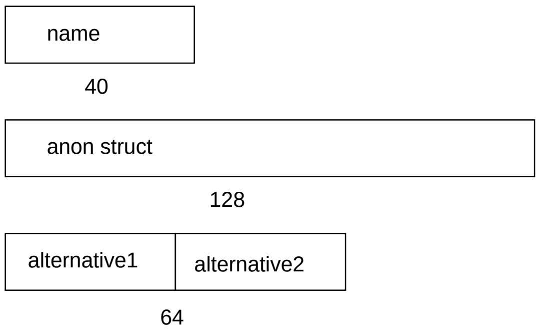
- Need to assume that two identifiers or two constants can't occur consecutively without some intervening separator, e.g. an operator, parenthesis, or language keyword. Otherwise there is no way to determine identifier or constant boundaries
- Need to use lookahead to distinguish integers from reals in the presence of operators that start with a dot. For example: `3.LT.N` (integer compare) and `3.E10` (real constant)
- Need to use lookahead to distinguish real constants from equality comparisons, For example `3.EQ.N` vs. `3.E10`
- Need to deal correctly with multiple dots. For example `3..NE..4.OR..2.LE..5`
- Need to be able to detect mal-formed constants, e.g. `3.4.5`

4. [20 marks] First map the innermost anonymous structure in the union



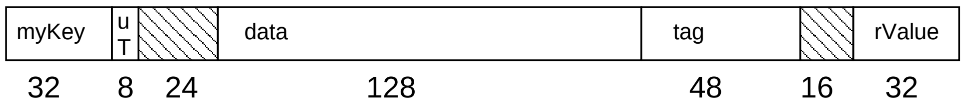
length: 128 alignment: 64

Next map the three alternatives in the union



length: 128 alignment: 64

Finally map the outer structure

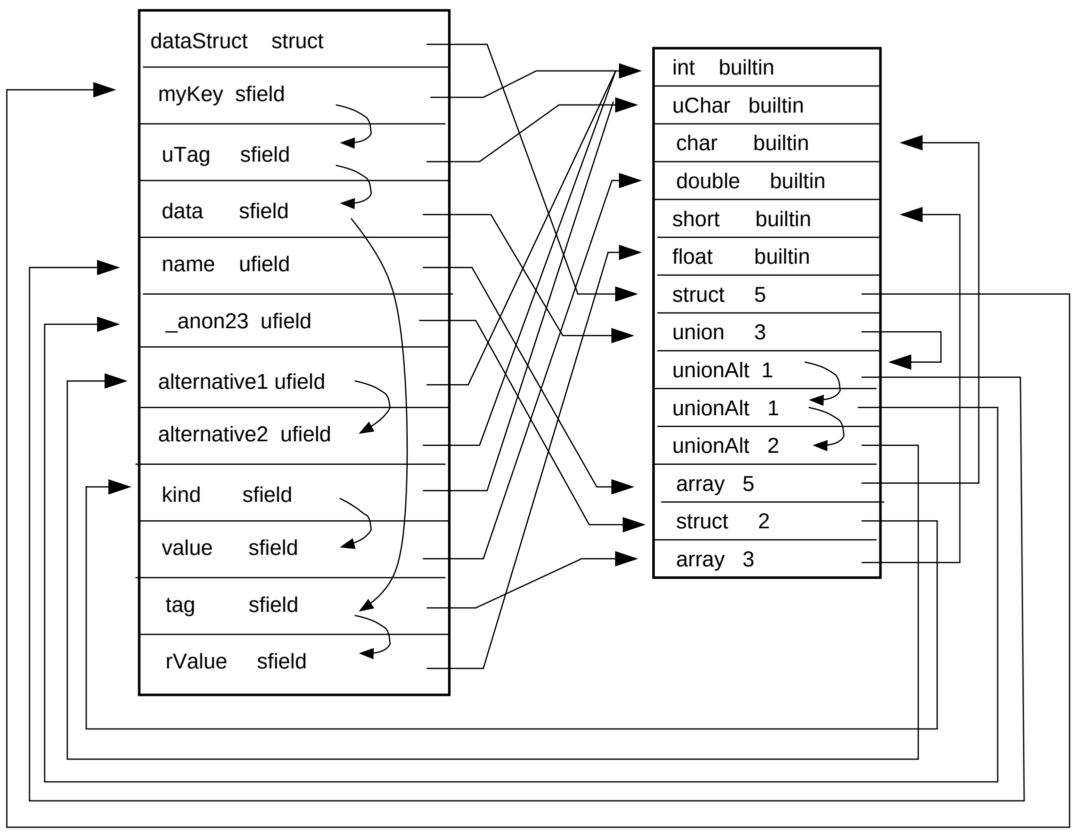


length: 288 (36 bytes) alignment: 64 (8 bytes)

Many solutions didn't handle the overlapping union fields correctly.

5. [15 marks]

The symbol and type tables should look something like this:



Most problems were incompleteness, missing links or incomplete handling of the union alternatives.