Plan for Today

Error recovery goals

Review panic mode error recovery for predictive parsers

Panic mode for LR parsers

Error recovery using error symbol in productions

Error Handling Goals

Provide program with a list of as many errors as possible

Provide USEFUL error messages
– appropriate line and position information
– guidance for fixing the error

Avoid infinite loops or recursion

Add minimal overhead to the processing of correct programs

Find “all” errors in program before translation begins

Predictive parser for Float Assignment Grammar

```c
void S() { switch (lookahead) {
    case ID: // the 2 tokens in the FIRST(StmList EOF)
        try { StmList(); match(EOF); } catch { panic(S); } break;
    default: panic(S); break;
}}
void StmList() { switch (lookahead) {
    case ID: // FIRST(Stm StmList) = { ID }
        try { Stm(); StmList(); } catch { panic(StmList); } break;
    case EOF: // FOLLOW(StmList) = { EOF }
        break;
    default: panic(StmList); break;
}}
void Stm() { switch (lookahead) {
    case ID: try { match(ID); match(ASSIGN); match(FLOAT);
        } catch { panic(Stm); } break;
    default: panic(Stm); break;
}}
```

Grammar 3.1 from Tiger book

{(0) S' -> S $
(1)S -> S ; S
(2)S -> id := E
(3)S -> print (L)
(4)E -> id
(5)E -> num
(6)E -> E + E
(7)E -> (S,E)
(8)L -> E
(9)L -> L, E
Error recovery using an error symbol

exp -> ( error )
exp -> error ; exp

Steps taken when error occurs
(0) generate error indicating expected token(s)
(1) pop off stack until have state with shift action for error token
(2) shift the error token
(3) throw away input tokens until hit token with non-error action
(4) resume parsing

Suggested Exercises

Show the Stack, Input, and Action table (see Figure 3.18 in handout) using the parse table on slides 5 and 6 where the parser is using panic mode recovery for the following inputs

:= b + c ; $
(d := 5 + 6, 3)$ $
(())$