Quiz 6

1. **What are the main steps of the shift-reduce parsing algorithm?**

   1. Construct the action-goto table from the given grammar
   2. Apply the shift-reduce parsing algorithm to construct the parse tree

2. **What makes difference between different types of shift-reduce parsing such as SLR, CLR, and LALR?**

   action-goto table

3. **Give the shift-reduce parsing algorithm?**

```
set ip to point to the first symbol of the input string w$
repeat forever
begin
        if action[top(stack), current-input(ip)] = shift(s) then begin
                push current-input(ip) then s on top of the stack
                advance ip to the next input symbol
        end
        else if action[top(stack), current-input(ip)] = reduce A → β then
        begin
                pop 2*| β | symbols off the stack;
                push A then goto[top(stack), A] on top of the stack;
                output the production A → β
        end
        else if action[top(stack), current-input(ip)] = accept then
                return
        else error()
end
```

4. **When the shift-reduce parsing algorithm repeat loop will stop?**

   When the action: result of the parser has current action and state as the input, **is "accept"**, the repeating of the loop will stop.

5. **Briefly compare between the following parsers: Simple LR (SLR), Canonical LR (CLR) and Lookahead LR (LALR)?**

   Simple LR (SLR): succeeds for the fewest grammars, but is the easiest to implement.

   Canonical LR: succeeds for the most grammars, but is the hardest to implement. It splits stateswhen necessary to prevent reductions that would get the parser stuck.

   Lookahead LR (LALR): succeeds for most common syntactic constructions used in programming languages, but produces LR tables much smaller than canonical LR.

6. Which parsing technique covers the largest class of grammars?

Canonical LR parser is.

It can manages most of the grammars. In the parsing techniques, it has the biggest cover range.

7. Briefly compare between top-down predictive parsers and bottom-up shift-reduce parsers?

Top-down predictive parser:

Left to right Leftmost-derivation Parser. As a working process, it constructs the parsing table. It starts only with the root non-terminal on the stack, and it ends when the stack is empty. It expands non-terminal. Uses pre-order traversal of the parse tree.

Bottom-up predictive parser:

Left to right Rightmost-derivation Parser. As a working process, it constructs the action-goto table. It ends only with the root non-terminal on the stack, and it starts with an empty stack. It reduces non-terminal. Uses post-order traversal of the parse tree.