MaxHs Grammar

module	→ `module` `ModId` *exports* `where` *body*
exports	→ ϵ
exports	→ `(`*exports1*`)`
exports1	→ *var*
exports1	→ *var* `,` *exports1*
body	→ {*decls*}
decls	→ *decl*
decls	→ *decl* `;` *decls*
decl	→ *gendecl*
decl	→ *funlhs rhs*
decl	→ *var rhs*
gendecl	→ *vars* `::` *type*
vars	→ *var*
vars	→ *var* `,` *vars*
type	→ *atype*
type	→ *atype* `->` *type*
atype	→ `ConId`
atype	→ `(`*types*`)`
atype	→ `[`*type*`]`
types	→ *type*
types	→ *type* `,` *types*
funlhs	→ *var args*
args	→ *var*
args	→ *var args*
rhs	→ `=` *exp*
rhs	→ `=` *exp* `where` {*decls*}
exp	→ *bit_or_exp* `::` *type*
exp	→ *bit_or_exp*
bit_or_exp	→ *bit_or_exp* `|` *bit_xor_exp*
bit_or_exp	→ *bit_xor_exp*
bit_xor_exp	→ *bit_xor_exp* `^` *bit_and_exp*
bit_xor_exp	→ *bit_and_exp*
bit_and_exp	→ *bit_and_exp* `&` *comp_exp*
bit_and_exp	→ *comp_exp*
comp_exp	→ *add_exp* `===` *add_exp*
comp_exp	→ *add_exp* `!==` *add_exp*
comp_exp	→ *add_exp* `<` *add_exp*
comp_exp	→ *add_exp* `>` *add_exp*
comp_exp	→ *add_exp* `<=` *add_exp*
comp_exp	→ *add_exp* `>=` *add_exp*
comp_exp	→ *add_exp*
add_exp	→ *add_exp* `+` *mul_exp*
add_exp	→ *add_exp* `-` *mul_exp*
add_exp	→ *mul_exp*
mul_exp	→ *mul_exp* `*` *pre_exp*
mul_exp	→ *mul_exp* `/` *pre_exp*
mul_exp	→ *pre_exp*
pre_exp	→ `-` *epx10*
pre_exp	→ *exp10*
exp10	→ `let` {*decls*} `in` *exp*
exp10	→ `if` *exp* `then` *exp* `else` *exp*
exp10	→ *fexp*
fexp	→ *aexp*
fexp	→ *fexp aexp*
aexp	→ *var*
aexp	→ *literal*
aexp	→ `(`*exps*`)`
aexp	→ `[`*exps*`]`
exps	→ *exp*
exps	→ *exp* `,` *exps*

LL grammar

module	→ `module` `ModId` *exports* `where` *body*
exports	→ ϵ
exports	→ `(`*var exports'*`)`
exports'	→ ϵ
exports'	→ `,` *var exports'*
body	→ {*decls*}
decls	→ ϵ
decls	→ *decl decls'*
decls'	→ ϵ
decls'	→ `;` *decl decls'*
decl	→ *var decl'*
decl'	→ *gendecl'*
decl'	→ *funlhs' rhs*
decl'	→ *rhs*
gendecl'	→ *vars'* `::` *type*
vars'	→ ϵ
vars'	→ `,` *vars'*
type	→ *atype type'*
type'	→ ϵ
type'	→ `->` *atype type'*
atype	→ `ConId`
atype	→ `(`*types*`)`
atype	→ `[`*type*`]`
types	→ *type types'*
types'	→ ϵ
types'	→ `,` *type types'*
funlhs'	→ *var args'*
args'	→ ϵ
args'	→ *var args'*
rhs	→ `=` *exp*
rhs	→ `=` *exp* `where` {*decls*}
exp	→ *bit_or_exp* `::` *type*
exp	→ *bit_or_exp*
bit_or_exp	→ *bit_xor_exp bit_or_exp'*
bit_or_exp'	→ ϵ
bit_or_exp'	→ `|` *bit_xor_exp bit_or_exp'*

bit_xor_exp	→ *bit_and_exp bit_xor_exp'*
bit_xor_exp'	→ ϵ
bit_xor_exp'	→ `^` *bit_and_exp bit_xor_exp'*
bit_and_exp	→ *comp_exp bit_and_exp'*
bit_and_exp'	→ ϵ
bit_and_exp'	→ `&` *comp_exp bit_and_exp'*
comp_exp	→ *add_exp comp_exp'*
comp_exp'	→ ϵ
comp_exp'	→ `===` *add_exp*
comp_exp'	→ `!==` *add_exp*
comp_exp'	→ `<` *add_exp*
comp_exp'	→ `>` *add_exp*
comp_exp'	→ `<=` *add_exp*
comp_exp'	→ `>=` *add_exp*
add_exp	→ *mul_exp add_exp'*
add_exp'	→ ϵ
add_exp'	→ `+` *mul_exp add_exp'*
add_exp'	→ `-` *mul_exp add_exp'*
mul_exp	→ *pre_exp mul_exp'*
mul_exp'	→ ϵ
mul_exp'	→ `*` *pre_exp mul_exp'*
mul_exp'	→ `/` *pre_exp mul_exp'*
pre_exp	→ `-` *exp*10
pre_exp	→ *exp*10
*exp*10	→ `let` {*decls*} `in` *exp*
*exp*10	→ `if` *exp* `then` *exp* `else` *exp*
*exp*10	→ *fexp*
fexp	→ *aexp fexp'*
fexp'	→ ϵ
fexp'	→ *aexp fexp'*
aexp	→ *var*
aexp	→ *literal*
aexp	→ `(`*exps*`)`
aexp	→ `[`*exps*`]`
exps	→ *exp exps'*
exps'	→ ϵ
exps'	→ `,` *exp exps'*