1. Конкретный синтаксис.

⟨*ident*⟩ ::= ⟨*nondigit*⟩
| ⟨*nondigit*⟩ ⟨*alphanum*⟩
| ⟨*nondigit*⟩ ⟨*alphanum*⟩ ⟨*ident*⟩

⟨*binop*⟩ ::= ‘+’ | ‘.-’ | ‘^’ | ‘*’ | ‘/’ | ‘==’ | ‘/= ’ | ‘<=’ | ‘<’ | ‘>’ | ‘>=’ | ‘|’ | ‘&&’

⟨*unop*⟩ ::= ’-’ | ’!’

⟨*num-lit*⟩ ::= number

⟨*bool-lit*⟩ ::= ’T’ | ’F’

⟨*lit*⟩ ::= ⟨*num-lit*⟩ | ⟨*bool-lit*⟩

⟨*expr*⟩ ::= ⟨*ident*⟩
| ⟨*lit*⟩
| ⟨*app*⟩
| ’if’ ⟨*expr*⟩ ’then’ ⟨*expr*⟩ ’else’ ⟨*expr*⟩
| ’let’ var ’=’ ⟨*expr*⟩ ’in’ ⟨*expr*⟩
| ⟨*expr*⟩ ⟨*binop*⟩ ⟨*expr*⟩
| ⟨*unop*⟩ ⟨*expr*⟩
| ’(’ ⟨*expr*⟩ ’)’

⟨*app*⟩ ::= ⟨*ident*⟩ ’ ’ ⟨*ident*⟩
| ⟨*ident*⟩ ’ ’ ⟨*lit*⟩
| ⟨*ident*⟩ ’ ’ ’(’ ⟨*expr*⟩’)’
| ’(’ ⟨*expr*⟩ ’)’ ⟨*ident*⟩
| ’(’ ⟨*expr*⟩ ’)’ ⟨*lit*⟩
| ’(’ ⟨*expr*⟩ ’)’ ’(’ ⟨*expr*⟩ ’)’

⟨*bind*⟩ ::= ⟨*ident*⟩ (⟨*arg*⟩ | ε) ’=’ ⟨*expr*⟩

⟨*arg*⟩ ::= ⟨*ident*⟩ | ⟨*ident*⟩ ’ ’ ⟨*arg*⟩

⟨*decl*⟩ ::= ⟨*bind*⟩ ’;’ ⟨*decl*⟩ | ⟨*bind*⟩ ’;’

Примеры.

1. Объявление функции/переменной.

```
f x = x;
n = 10;
```

2. Использование условного оператора `if`

```
f x = if x == 0 then 10 else 20 + x;
```

3. Использование `let`-связывания.

```
f y = let x = 10 * y in x ^ x;
```

4. Вызов функции

```
f x = x;
g = f 10;

ff x y = x + y;
gg = ff 10 20;
```

2. Абстрактный синтаксис

Представлен в виде АСД.

```
type Ast = [Decl]

data Decl = BindDecl Bind

data Bind = Bind Name [Name] Expr

data Expr = Var Name
          | Lit Lit
          | App Expr Expr
          | If Expr Expr Expr
          | Let Name Expr Expr
          | UnOp UnOperator Expr
          | BinOp BinOperator Expr Expr

data Lit = ILit Integer | BLit Bool
```