(** * IMP Syntax *)

Require Import ZArith.
Require Import String.

Open Scope string_scope.
Open Scope Z_scope.

Inductive binop : Set :=
| Add | Sub | Mul | Div | Mod | Lt  | Lte | Conj | Disj.

Inductive expr : Set :=
| Int : Z -> expr  | Var : string -> expr  |

Coercion Int : Z >> expr.
Coercion Var : string >> expr.

Notation "X [+] Y" := (BinOp Add X Y) (at level 51, left associativity).
Notation "X [−] Y" := (BinOp Sub X Y) (at level 51, left associativity).
Notation "X [*] Y" := (BinOp Mul X Y) (at level 50, left associativity).
Notation "X [/] Y" := (BinOp Div X Y) (at level 50, left associativity).
Notation "X [%] Y" := (BinOp Mod X Y) (at level 50, left associativity).
Notation "X [<] Y" := (BinOp Lt X Y) (at level 52).
Notation "X [<=] Y" := (BinOp Lte X Y) (at level 52).
Notation "X [&&] Y" := (BinOp Conj X Y) (at level 53, left associativity).
Notation "X [||] Y" := (BinOp Disj X Y) (at level 54, left associativity).

Inductive stmt : Set :=
| Nop : stmt  |
| Assign : string -> expr -> stmt |
| Seq : stmt -> stmt -> stmt |
| Cond : expr -> stmt -> stmt |
| While : expr -> stmt -> stmt. |

Notation "nop" := (Nop) (at level 60).
Notation "X [−] Y" := (Assign X Y) (at level 60).
Notation "X [−] Y" := (Seq X Y) (at level 61).
Notation "if X [−] Y" := (Cond X Y) (at level 60).
Notation "while X [−] Y" := (While X Y) (at level 60).