```
HEADER.html:   HTML document, ASCII text
Listing.pdf:   PDF document, version 1.4
Listing.ps:    PostScript document text conforming DSC level 3.0
Makefile:      ASCII text
Makefile.dep:  ASCII text
RCS:           directory
absyn.cmi:     OCaml interface file (.cmi) (Version 026)
absyn.mli:     ASCII text
dumper.cmi:    OCaml interface file (.cmi) (Version 026)
dumper.cmx:    OCaml native object file (.cmx) (Version 026)
dumper.ml:     ASCII text
dumper.mli:    ASCII text
dumper.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
etc.cmi:       OCaml interface file (.cmi) (Version 026)
etc.cmx:       OCaml native object file (.cmx) (Version 026)
etc.ml:        ASCII text
etc.mli:       ASCII text
etc.o:         ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
interp.cmi:    OCaml interface file (.cmi) (Version 026)
interp.cmx:    OCaml native object file (.cmx) (Version 026)
interp.ml:     ASCII text
interp.mli:    ASCII text
interp.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
main.cmi:      OCaml interface file (.cmi) (Version 026)
main.cmx:      OCaml native object file (.cmx) (Version 026)
main.ml:       ASCII text
main.o:        ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
mb-programs.d: symbolic link to `/afs/cats.ucsc.edu/courses/cse112-wm/Assignmen
mbinterp:      ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
namedefs.d:    directory
parser.cmi:    OCaml interface file (.cmi) (Version 026)
parser.cmx:    OCaml native object file (.cmx) (Version 026)
parser.ml:     ASCII text
parser.mli:    ASCII text
parser.mly:    ASCII text
parser.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
parser.output: ASCII text
scanner.cmi:   OCaml interface file (.cmi) (Version 026)
scanner.cmx:   OCaml native object file (.cmx) (Version 026)
scanner.ml:    ASCII text
scanner.mll:   ASCII text
scanner.o:     ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
tables.cmi:    OCaml interface file (.cmi) (Version 026)
tables.cmx:    OCaml native object file (.cmx) (Version 026)
tables.ml:     ASCII text
tables.mli:    ASCII text
tables.o:      ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripp
using:         ASCII text
```

```
::::::::::::::
absyn.mli.defs
::::::::::::::
type linenr = int
type ident = string
type label = string
type oper = string
type memref = Arrayref of ident * expr | Variable of ident
and expr =
    Number of float
  | Memref of memref
  | Unary of oper * expr
  | Binary of oper * expr * expr
and relexpr = Relexpr of oper * expr * expr
type printable = Printexpr of expr | String of string
type stmt =
    Dim of ident * expr
  | Let of memref * expr
  | Goto of label
  | If of relexpr * label
  | Print of printable list
  | Input of memref list
type progline = linenr * label option * stmt option
type program = progline list
```

```
::::::::::::::
dumper.mli.defs
::::::::::::::
val quote : string -> string
val join : string -> string -> string -> string list -> string
val string_of_option : ('a -> string) -> 'a option -> string
val string_of_ctor : string -> string list -> string
val string_of_list : ('a -> string) -> 'a list -> string
val string_of_printable : Absyn.printable -> string
val string_of_memref : Absyn.memref -> string
val string_of_expr : Absyn.expr -> string
val string_of_relexpr : Absyn.relexpr -> string
val string_of_stmt : Absyn.stmt -> string
val dump_progline : int * string option * Absyn.stmt option -> unit
val dump_program : Absyn.program -> unit
::::::::::::::
dumper.ml.defs
::::::::::::::
val quote : string -> string
val join : string -> string -> string -> string list -> string
val string_of_option : ('a -> string) -> 'a option -> string
val string_of_ctor : string -> string list -> string
val string_of_list : ('a -> string) -> 'a list -> string
val string_of_printable : Absyn.printable -> string
val string_of_memref : Absyn.memref -> string
val string_of_expr : Absyn.expr -> string
val string_of_relexpr : Absyn.relexpr -> string
val string_of_stmt : Absyn.stmt -> string
val dump_progline : int * string option * Absyn.stmt option -> unit
val dump_program : Absyn.program -> unit
```

```
::::::::::::::
etc.mli.defs
::::::::::::::
val warn : string list -> unit
val die : string list -> unit
val syntax_error : Lexing.position -> string list -> unit
val usage_exit : string list -> unit
val read_number : unit -> float
val int_of_round_float : float -> int
::::::::::::::
etc.ml.defs
::::::::::::::
val execname : string
val exit_status_ref : int ref
val quit : unit -> unit
val eprint_list : string list -> unit
val warn : string list -> unit
val die : string list -> unit
val syntax_error : Lexing.position -> string list -> unit
val usage_exit : string list -> unit
val buffer : string list ref
val read_number : unit -> float
val int_of_round_float : float -> int
```

```
::::::::::::::
interp.mli.defs
::::::::::::::
val want_dump : bool ref
val interpret_program : Absyn.program -> unit
::::::::::::::
interp.ml.defs
::::::::::::::
exception Unimplemented of string
val no_expr : string -> 'a
val no_stmt : string -> 'a -> 'b
val want_dump : bool ref
val eval_expr : Absyn.expr -> float
val eval_memref : Absyn.memref -> float
val interpret : Absyn.program -> unit
val interp_stmt : Absyn.stmt -> Absyn.program -> unit
val interp_print : Absyn.printable list -> Absyn.program -> unit
val interp_input : Absyn.memref list -> Absyn.program -> unit
val interpret_program : Absyn.program -> unit
```

```
::::::::::::::
main.ml.defs
::::::::::::::
val interpret_source : string -> unit
```

```
::::::::::::::
parser.mli.defs
::::::::::::::
type token =
    RELOP of string
  | EQUAL of string
  | ADDOP of string
  | MULOP of string
  | POWOP of string
  | IDENT of string
  | NUMBER of string
  | STRING of string
  | COLON
  | COMMA
  | LPAR
  | RPAR
  | LSUB
  | RSUB
  | EOL
  | EOF
  | DIM
  | LET
  | GOTO
  | IF
  | PRINT
  | INPUT
val program : (Lexing.lexbuf -> token) -> Lexing.lexbuf -> Absyn.program
::::::::::::::
parser.ml.defs
::::::::::::::
type token =
    RELOP of string
  | EQUAL of string
  | ADDOP of string
  | MULOP of string
  | POWOP of string
  | IDENT of string
  | NUMBER of string
  | STRING of string
  | COLON
  | COMMA
  | LPAR
  | RPAR
  | LSUB
  | RSUB
  | EOL
  | EOF
  | DIM
  | LET
  | GOTO
  | IF
  | PRINT
  | INPUT
val linenr : unit -> int
val syntax : unit -> unit
val yytransl_const : int array
val yytransl_block : int array
val yylhs : string
```

```
val yylen : string
val yydefred : string
val yydgoto : string
val yysindex : string
val yyrindex : string
val yygindex : string
val yytablesize : int
val yytable : string
val yycheck : string
val yynames_const : string
val yynames_block : string
val yyact : (Parsing.parser_env -> Obj.t) array
val yytables : Parsing.parse_tables
val program : (Lexing.lexbuf -> token) -> Lexing.lexbuf -> Absyn.program
```

```
::::::::::::::
scanner.ml.defs
::::::::::::::
val lexerror : Lexing.lexbuf -> unit
val newline : Lexing.lexbuf -> unit
val lexeme : Lexing.lexbuf -> string
val __ocaml_lex_tables : Lexing.lex_tables
val token : Lexing.lexbuf -> Parser.token
val __ocaml_lex_token_rec : Lexing.lexbuf -> int -> Parser.token
```

```
::::::::::::::
tables.mli.defs
::::::::::::::
type variable_table_t = (string, float) Hashtbl.t
type array_table_t = (string, float array) Hashtbl.t
type unary_fn_table_t = (string, float -> float) Hashtbl.t
type binary_fn_table_t = (string, float -> float -> float) Hashtbl.t
type label_table_t = (string, Absyn.program) Hashtbl.t
val variable_table : variable_table_t
val array_table : array_table_t
val unary_fn_table : unary_fn_table_t
val binary_fn_table : binary_fn_table_t
val label_table : label_table_t
val init_label_table : Absyn.program -> unit
val dump_label_table : unit -> unit
::::::::::::::
tables.ml.defs
::::::::::::::
type variable_table_t = (string, float) Hashtbl.t
type array_table_t = (string, float array) Hashtbl.t
type unary_fn_table_t = (string, float -> float) Hashtbl.t
type binary_fn_table_t = (string, float -> float -> float) Hashtbl.t
type label_table_t = (string, Absyn.program) Hashtbl.t
val variable_table : variable_table_t
val array_table : array_table_t
val unary_fn_table : unary_fn_table_t
val binary_fn_table : binary_fn_table_t
val label_table : label_table_t
val init_label_table : Absyn.program -> unit
val dump_label_table : unit -> unit
```