Installation fix and sample hello.y
hello.y:

```
%{

#include <stdio.h>
#include <stdlib.h>
int yylex(void);
int yyerror(const char *s);

%}

%token HI BYE

%%

program:
         hi bye
        ;

hi:
        HI     { printf("Hello World\n");   }
        ;
bye:
        BYE    { printf("Bye World\n"); exit(0); }
         ;
```

Lab finished lab8.y:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define YYDEBUG 1
int yylex();
int yyerror(char *s);
%}

%token BEGIN
%token END
%token INTEGER
%token STRING
%token BOOLEAN
%token List
%token append
%token REMOVE
%token length
%token Struct
%token True
```

```
%token False
%token read
%token PRINT
%token IF
%token ELSE
%token FOR

%token IDENTIFIER

%token NEW_LINE
%token AND
%token OR
%token COLON
%token SEMI_COLON
%token COMA
%token DOT
%token PLUS
%token MINUS
%token MULTIPLY
%token DIVISION
%token LEFT_ROUND_PARANTHESIS
%token RIGHT_ROUND_PARANTHESIS
%token LEFT_SQUARE_PARANTHESIS
%token RIGHT_SQUARE_PARANTHESIS
%token LEFT_CURLY_PARANTHESIS
%token RIGHT_CURLY_PARANTHESIS
%token GREATER_THAN
%token GREATER_OR_EQUAL_THAN
%token LESS_OR_EQUAL_THAN
%token LESS_THAN
%token ASSIGNMENT
%token EQUAL
%token DIFFERENT
%token NEGATION

%%

program : BEGIN decllist cmpdstmt END {printf("\n start -> decllist cmpdstmt <-
end\n");}
          ;
decllist : declaration | declaration decllist
          ;
declaration :  type IDENTIFIER {printf("\n type IDENTIFIER\n");}
         ;
type : type1 | arraydecl
        ;
type1 : INTEGER | STRING | BOOLEAN | arraydecl

arraydecl : List LESS_THAN type1 GREATER_THAN
```

```
cmpdstmt : stmt {printf("\n stmt\n");} | stmt cmpdstmt {printf("\n stmt cmpdstmt\n");}
      ;
stmt :  simplstmt NEW_LINE {printf("\n simplstmt NEW_LINE\n");} | structstmt
{printf("\n structstmt\n");}
     ;
simplstmt :  assignstmt {printf("\n assignstmt\n");} | iostmt {printf("\n iostmt\n");}
| declaration {printf("\n declaration\n");}
      ;
structstmt :  IFstmt {printf("\n IFstmt\n");} | FORstmt {printf("\n FORstmt\n");} |
cmpdstmt
       ;
IFstmt :  IF boolean_condition COLON LEFT_CURLY_PARANTHESIS cmpdstmt
RIGHT_CURLY_PARANTHESIS ELSEstmt {printf("\n IFstmt boolean_condition: {cmpdstmt}\n");}
       ;
ELSEstmt : /*empty*/ | ELSE COLON LEFT_CURLY_PARANTHESIS cmpdstmt
RIGHT_CURLY_PARANTHESIS
       ;
FORstmt :  FOR FORheader COLON LEFT_CURLY_PARANTHESIS cmpdstmt RIGHT_CURLY_PARANTHESIS
       ;
FORheader :  assignstmt boolean_condition assignstmt
       ;
assignstmt :  IDENTIFIER ASSIGNMENT expression {printf("\n IDENTIFIER ASSIGNMENT
expression\n");} | IDENTIFIER ASSIGNMENT iostmt {printf("\n IDENTIFIER ASSIGNMENT
iostmt\n");}
       ;
expression : arithmetic2 arithmetic1
       ;
arithmetic1 : PLUS arithmetic2 arithmetic1 | MINUS arithmetic2 arithmetic1 | /*Empty*/
        ;
arithmetic2 : multiply2 multiply1
        ;
multiply1 : MULTIPLY multiply2 multiply1 | DIVISION multiply2 multiply1 | /*Empty*/
       ;
multiply2 : LEFT_ROUND_PARANTHESIS expression RIGHT_ROUND_PARANTHESIS | IDENTIFIER
           ;
iostmt :  read LEFT_ROUND_PARANTHESIS type RIGHT_ROUND_PARANTHESIS | PRINT
LEFT_ROUND_PARANTHESIS IDENTIFIER RIGHT_ROUND_PARANTHESIS
      ;
condition : expression GREATER_THAN expression {printf("\n expression >
expression\n");} |
      expression GREATER_OR_EQUAL_THAN expression {printf("\n expression >=
expression\n");} |
      expression LESS_THAN expression {printf("\n expression < expression\n");} |
      expression LESS_OR_EQUAL_THAN expression {printf("\n expression <=
expression\n");} |
      expression EQUAL expression {printf("\n expression == expression\n");} |
      expression NEGATION ASSIGNMENT expression {printf("\n expression !=
expression\n");}
```

```
     ;
boolean_condition : condition boolean_cond_temp
            ;
boolean_cond_temp : /*Empty*/ | AND boolean_condition | OR boolean_condition
           ;


%%

int yyerror(char *s)
{
  printf("%s\n", s);
  return -1;
}

extern FILE *yyin;

int main(int argc, char **argv)
{
  if (argc > 1)
    yyin = fopen(argv[1], "r");
  if ( (argc > 2) && ( !strcmp(argv[2], "-d") ) )
    yydebug = 1;
  if ( !yyparse() )
    fprintf(stderr,"\t Awsm\n");

     return 0;
}
```