Compiler (Fall 2018) Programming Assignment 2

Program 2: an LR parser using Flex and Bison
Due: before class, Dec. 11, 2018

Given the following ATG representing a Robot Move, you are to use **Flex** and **Bison** to translate it and print out the final position of the Robot.

1. L → Seq {print (x, y) } where x=Seq.x, y=Seq.y
2. Seq → **begin** { seq.x=0; seq.y=0}
3. Seq→ Seq Instr { Seq.x=Seq_1.x + Inst.dx; Seq.y=Seq_1.y + Inst.dy}
4. Instr→ **east** { Instr.dx=1; Instr.dy=0}
5. Instr→ **north** { Instr.dx=0; Instr.dy=1}
6. Instr→ **west** { Instr.dx=-1; Instr.dy=0}
7. Instr→ **south** { Instr.dx=0; Instr.dy=-1}

Where Synthesized: Seq.x, Seq.y, Instr.dx, Inst.dy
Test data:
begin east east north west south
begin south south west west north south south
begin
begin southern

Good documentation is required. Pleas upload your program to the portal and also turn in a **hard copy** with execution snapshots when running the above test data.
//try to include something like this in .y file

```
%token BEGIN EAST ……
%union{
      struct
      { int x;
        int y;
      } coordinate;
…….
}
%type <coordinate>   Seq     Instr
//use     print ($1. x, $1.y),   $$.x=1, $$.y=0
```

The following is a fragment of Flex and Bison from a past programming assignment.

```
/* Sample Lex Yacc programs */

/***************** sample file: xx.l
**************************************/
/*  定義區  */
%{
#include <stdio.h>
#include <stdlib.h>
#include "xx.tab.h" /*  會自動生成的頭檔  */
void yyerror(char *); /*  偵錯函數  */
%}
/* Regular  定義區  */
…..
letter [A-Za-z]
id {letter}({letter}|{digit})*
number {digit}+
……..
/*  規則區  */
%%
">=" return GE;
"<=" return LE;
……..
"true" return TRUE;
"false" return FALSE;
……..
{id} { yylval.string = strdup(yytext);
return VARIABLE;}
{number} { yylval.string = strdup(yytext);
return INTEGER; }
%%
/*  偵錯函數  */
yywrap(void)
{
return 1;
}
/******************* file : xx.y ***********************************/
*  初始定義區  */
```

```
%{
#include <ctype.h>
#include <stdio.h> /* C 內建函數 */
#include <string.h> /* 用來幫數比較字串函數 */
void add_to_table(char *); /* 將變數加入 Symbol Table 之函數 */
void yyerror(char *); /* 偵錯函數 */
int check(char *); /* 檢查是否與 Symbol Table 內的變數重複之函數 */
char * symbol_table[100]; /* Symbol Table 的指標型 Array */
int NumberOfID; /* 計算 Symbol Table 內的個數值 */
int i; /* 計數變數 */
%}
/* 保留字定義區 */
%token BASIC WRITE TRUE FALSE BLEFT BRIGHT AND OR NOT SLEFT
SRIGHT ASSIGN
…….
/* 創造變數 */
%union
{
char *string;
}
/* 指定變數的 Type */
%token <string> INTEGER
%token <string> VARIABLE
/* 規則定義區 */
%%
program:
block;
block:
BLEFT decls stmts BRIGHT
|;
decl: /* 若 Type 屬於 Basic 的話，將之存入 Symbol Table */
type VARIABLE { add_to_table($2); }; ……..
bool:
bool OR join { printf("Or\n"); } /* 若為||敘述，印出 OR 字樣 */
| join;
……….
%%
/* 函數定義區 */
```

```
/* 偵錯函數 */
void yyerror(char *s)
{
fprintf(stderr, "%s\n", s);
}
/* 主程式函數 */
int main(void)
{
NumberOfID = 0; /* 放入初始值 */
yyparse();
return 0;
}
/* 將變數加入 Symbol Table 之函數 */
void add_to_table(char *temp)
{ ……..
…….}
/* 檢查是否與 Symbol Table 內的變數重複之函數 */
int check(char *temp)
{….
}……
```