Compiler Final Project

Group 14　　組員：00757019 沈奎宏　　00757046 黃子軒

壹、組員分配的工作

組員	分工
沈奎宏	題目發想、詞彙分析、文法設計、程式撰寫、程式 Debug 與測試
黃子軒	查詢資料、詞彙分析、文法設計、程式撰寫、製作 PPT

貳、專題中遭遇的困難

A. 負整數會影響四則運算結果

我們在測試各式運算時，發現負整數的加入會導致程式出錯，讓程式無法精準判別正負號，於是我們加入「負整數之負號必須加在括弧內」的限制，藉此可以和 SUB(-)做出區隔，也可以讓整體運算回歸清晰且正確。

B. 只能放正整數的運算

我們在測試階乘、最大公因數、最小公倍數時，發現負整數的加入並不符合數學邏輯，於是我們將只能放入正整數 token 的位置換成 NUMBER，如此就能達到「僅有正整數能出現的位置是正整數」的效果，確保整體運算正確。

C. 最終印出的答案型態不能更動

我們在測試 POW(^)時，發現次方可以為負數，而計算機也確實會進行次方

為負數的運算，但可惜的是 Flex&Bison 編譯器最終只能印出整數部分，小數點後的值無法顯示出來。我們嘗試從.tab.c 和.tab.h 檔去更改答案型態，可每次執行時，仍會更新成原本的 int 型態。

參、執行結果

```
Rules and Restrictions
    1. 負數必須打在括號中
    2. 次方可以為負，但編譯器只能處理int，故只會顯示 0，e.g. 2^(-1) = 0
    3. 除法同上原因，若結果為小數也僅能顯示整數
    4. 階乘、最大公因數、最小公倍數，只能讀取正整數，e.g. 5!

Test Cases:
    14
    (-14)
    1 + 4 - 7
    6 * 4 / 2
    2 * (3 + 4)
    10 % 3 * 14
    14 / 7 / 2
    4 + 2 * (14 % 9)
    2 ^ 10
    (-2) ^ 5
    5 ^ 2 ^ 1
    14 ^ 0 ^ 10
    3 ^ 2 ^ 3
    3 ^ (2 ^ 3)
    (2 + 3) * 4
    162 / 3 ^ 4 * 2
    ((14 / 3) ^ 4) / 128 + 25 % 13

    0!
    1!
    5!
    3! ^ 2
    3 ^ 2!
    3! + 2! - 5! * 2! / 4!

    [100, 110]
    {110, 100}
    [144, 36]
    {36, 144}
```

```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\h5222> cd D:\Desktop\海海人生\大三下編譯器\Homework\FinalProject\Group14FinalProject
(base) PS D:\Desktop\海海人生\大三下編譯器\Homework\FinalProject\Group14FinalProject> flex finalToken.l
(base) PS D:\Desktop\海海人生\大三下編譯器\Homework\FinalProject\Group14FinalProject> bison -d finalGrammar.y
(base) PS D:\Desktop\海海人生\大三下編譯器\Homework\FinalProject\Group14FinalProject> gcc finalGrammar.tab.c lex.yy.c -lfl
(base) PS D:\Desktop\海海人生\大三下編譯器\Homework\FinalProject\Group14FinalProject> a
14
=14
(-14)
=-14
1 + 4 - 7
=-2
6 * 4 / 2
=12
2 * (3 + 4)
=14
10 % 3 * 14
=14
14 / 7 / 2
=1
4 + 2 * (14 % 9)
=14
2 ^ 10
=1024
(-2) ^ 5
=-32
5 ^ 2 ^ 1
=25
14 ^ 0 ^ 10
=1
3 ^ 2 ^ 3
=729
3 ^ (2 ^ 3)
=6561
(2 + 3) * 4
=20
162 / 3 ^ 4 * 2
=4
((14 / 3) ^ 4) / 128 + 25 % 13
=14
0!
=1
1!
=1
5!
=120
3! ^ 2
=36
3 ^ 2!
=9
3! + 2! - 5! * 2! / 4!
=-2
[100, 110]
=1100
{110, 100}
=10
[144, 36]
=144
{36, 144}
=36
```

肆、程式碼電子檔

finalToken.l

finalToken.l finalGrammar.y

D: > Desktop > 海海人生 > 大三下編譯器 > Homework > FinalProject > Group14FinalProject > finalToken.l

```
%{
#include "finalGrammar.tab.h"
%}

%%

"+"     {return ADD;}
"-"     {return SUB;}
"*"     {return MUL;}
"/"     {return DIV;}
"|"     {return ABS;}

"%"     {return MOD;}
"^"     {return POW;}
"!"     {return FAC;}

"("     {return FRONT;}
")"     {return BACK;}

","     {return COMMA;}

"["     {return FLCM;}
"]"     {return BLCM;}

"{"     {return FGCD;}
"}"     {return BGCD;}

[0-9]+  {yylval=atoi(yytext);return NUMBER;}
\n      {return EOL;}
[ \t]   {/*空白忽略*/}
.   {printf("非法字符 %c\n",*yytext);}

%%
```

finalGrammer.y

finalToken.l finalGrammar.y

D: > Desktop > 海海人生 > 大三下編譯器 > Homework > FinalProject > Group14FinalProject > finalGrammar.y

```
%{
#include <stdio.h>
#include <math.h>
%}

%token NUMBER
%token ADD SUB MUL DIV
%token MOD
%token POW FAC
%token FRONT BACK
%token COMMA
%token FLCM BLCM FGCD BGCD
%token ABS
%token EOL

%%

calclist:/*註解*/
  |calclist exp EOL{printf ("=%d\n",$2);}
  ;

exp:factor {$$ = $1;}
  |exp ADD factor{$$=$1+$3;}
  |exp SUB factor{$$=$1-$3;}
  ;

factor:term {$$=$1;}
  |factor MUL term{$$=$1*$3;}
  |factor DIV term{$$=$1/$3;}
  |factor MOD term{$$=$1%$3;}
  |FGCD NUMBER COMMA NUMBER BGCD{$$=gcd($2, $4);}
  |FLCM NUMBER COMMA NUMBER BLCM{$$=lcm($2, $4);}
  ;

term:power {$$=$1;}
  |term POW power{$$=pow($1, $3);}
  ;
```

```
power:factorial {$$=$1;}
  |NUMBER FAC{$$=fac($1);}
//  |FGCD NUMBER BACK FAC{$$=fac($2);}
  ;

factorial:paren {$$=$1;}
  |FRONT exp BACK{$$=$2;}
  |FRONT SUB exp BACK{$$=-$3;}
  ;

paren:NUMBER {$$=$1;}
  |ABS factorial {$$=$2>=0?$2:-$2;}
  ;
%%

fac(int num)
{
  int i;
  int factorialSum = 1;
  for(i = 1; i <= num; i++)
    factorialSum *= i;
  return factorialSum;
}

int gcd(int numB, int numS)
{
  int temp = numB;
  numB = (numB  > numS) ? numB : numS;
  numS = (numB == numS) ? temp : numS;

  int remainder;
  while(numS != 0)
  {
    remainder = numB % numS;
    numB = numS;
    numS = remainder;
  }
  return numB;
```

```
}

int lcm(int numB, int numS)
{
  return (numB * numS / gcd(numB, numS));
}

main(int argc,char **argv){
  yyparse();
}

yyerror(char *s)
{
 fprintf(stderr,"error:%s\n",s);
}
```