Compiler final project

00957202 吳秉宸
00957205 吳翊揩
00853029 張正德

目錄

1 簡介

2 專案程式碼.l

3 專案程式碼.y

4 Demo & 測試結果

5 專案遇到的問題

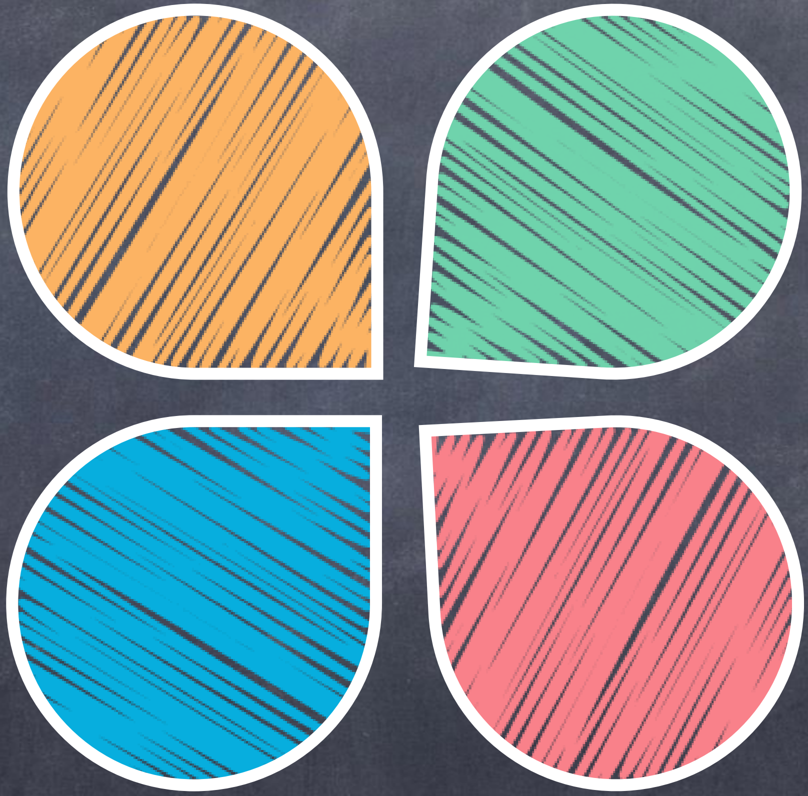
簡介

Compiler final project

我們做的是多項式的運算，可以做連續的加減與微分，並可以讓使用者宣告自定義的多項式式子的變數，其中使用者自定義的變數都可以暫存在記憶體中，重複定義會覆蓋掉原本的變數值。

而一個多項式的式子中可以有三個變數x,y,z，使用者可以進行賦值，會暫存在記憶體當中，且每次賦值都會輸出目前的x,y,z代入各個多項式計算出的結果(其中output會自動進行整合、排序，將次方大的優先輸出，若次方數一樣則按照x,y,z順序輸出)。

技術應用



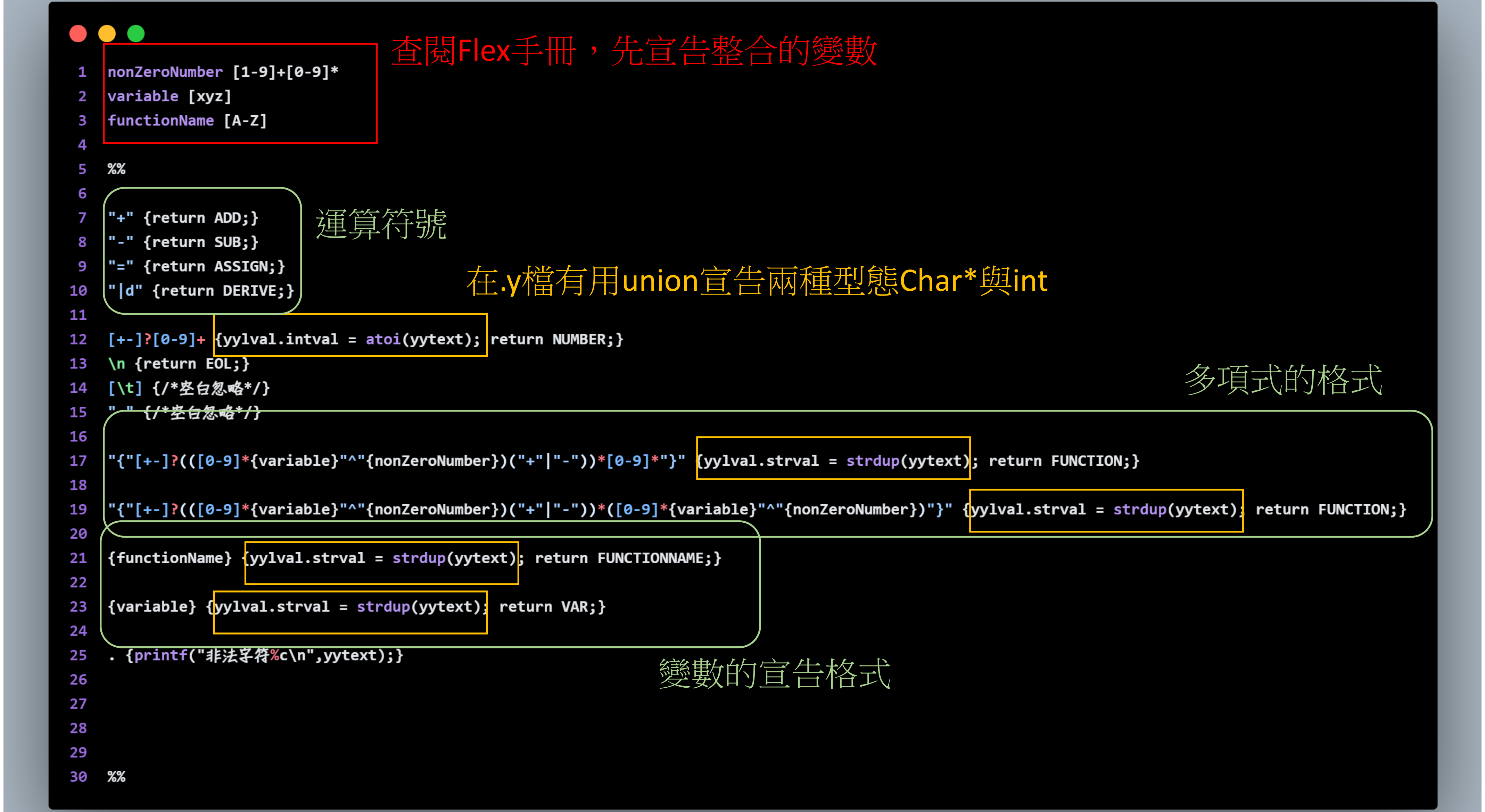
Flex(.l)

Action

Bison(.y)

C Language
&資料結構

專案程式碼.1



```
nonZeroNumber [1-9]+[0-9]*
variable [xyz]
functionName [A-Z]

%%

"+" {return ADD;}
"-" {return SUB;}
"=" {return ASSIGN;}
"|d" {return DERIVE;}

[+-]?[0-9]+ {yylval.intval = atoi(yytext); return NUMBER;}
\n {return EOL;}
[\t] {/*空白忽略*/}
" " {/*空白忽略*/}

"{"[+-]?(([0-9]*{variable}"^"{nonZeroNumber})("+"|"-"))*[0-9]*"}" {yylval.strval = strdup(yytext); return FUNCTION;}

"{"[+-]?(([0-9]*{variable}"^"{nonZeroNumber})("+"|"-"))*([0-9]*{variable}"^"{nonZeroNumber})"}" {yylval.strval = strdup(yytext); return FUNCTION;}

{functionName} {yylval.strval = strdup(yytext); return FUNCTIONNAME;}

{variable} {yylval.strval = strdup(yytext); return VAR;}

. {printf("非法字符%c\n",yytext);}




%%
```

查閱Flex手冊，先宣告整合的變數

運算符號

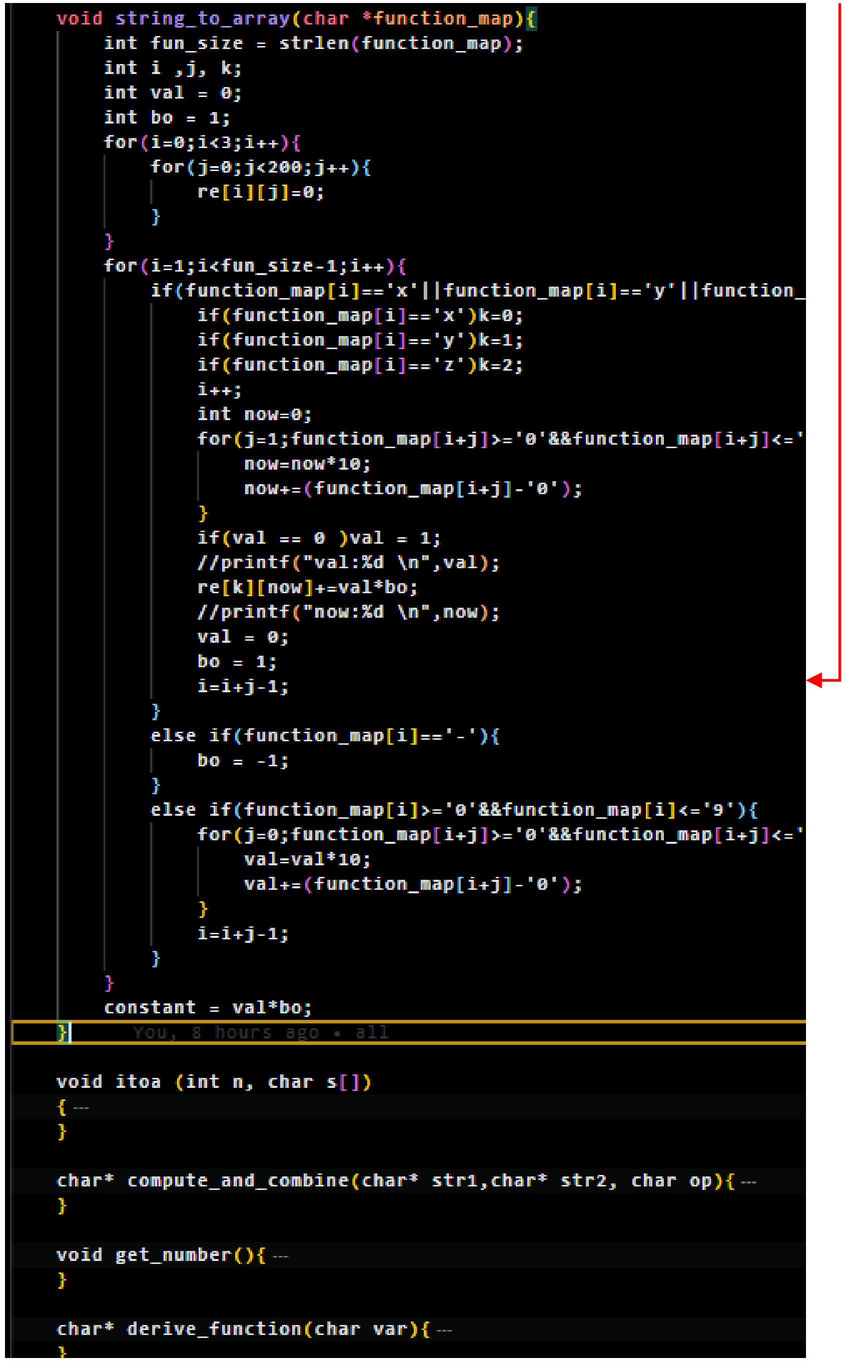
在.y檔有用union宣告兩種型態Char*與int

多項式的格式

變數的宣告格式

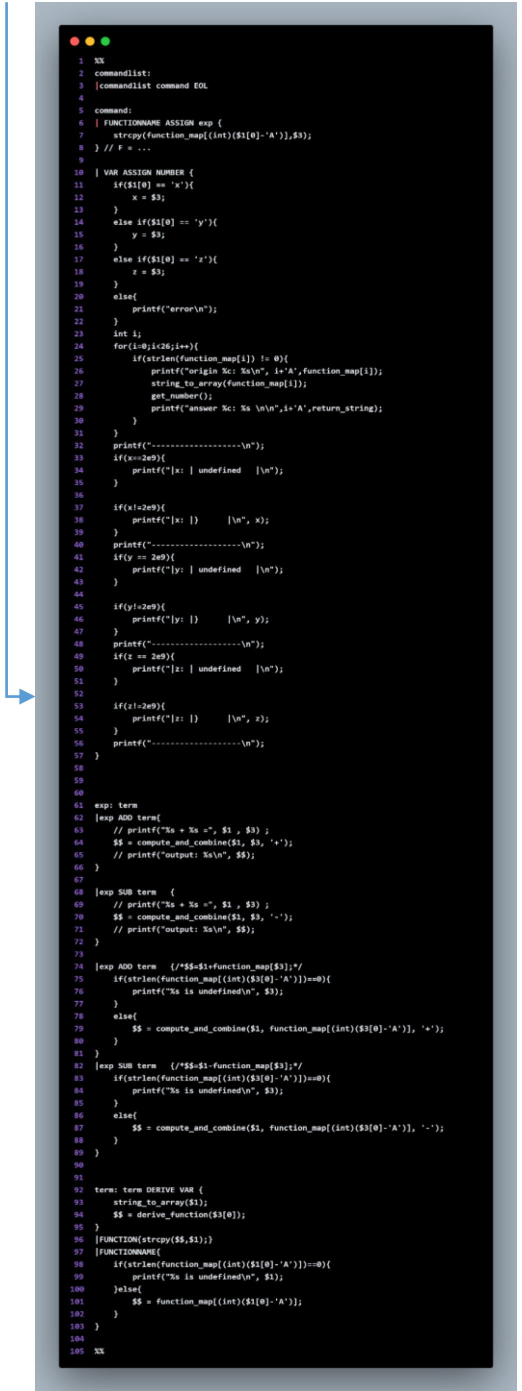
專案程式碼. y

研究出Bison如何自定義c的函式，結構化的處理程式

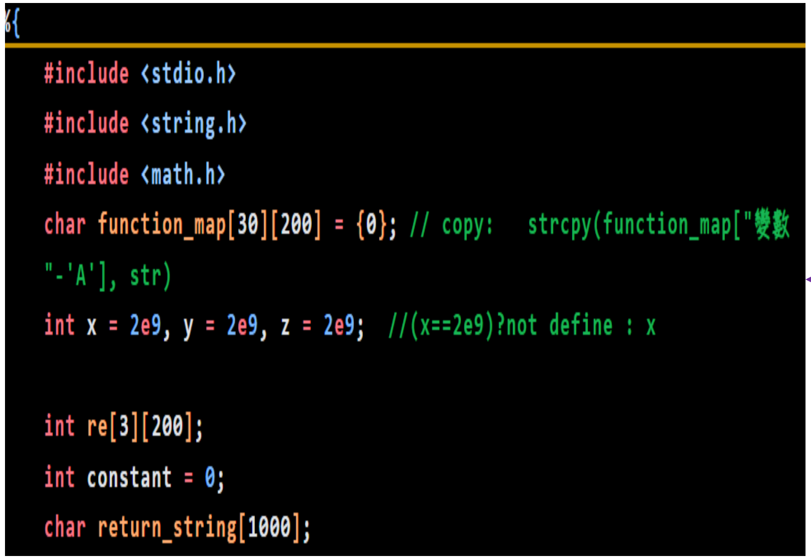


```
void string_to_array(char *function_map){
    int fun_size = strlen(function_map);
    int i ,j, k;
    int val = 0;
    int bo = 1;
    for(i=0;i<3;i++){
        for(j=0;j<200;j++){
            re[i][j]=0;
        }
    }
    for(i=1;i<fun_size-1;i++){
        if(function_map[i]=='x'||function_map[i]=='y'||function_
            if(function_map[i]=='x')k=0;
            if(function_map[i]=='y')k=1;
            if(function_map[i]=='z')k=2;
            i++;
            int now=0;
            for(j=1;function_map[i+j]>='0'&&function_map[i+j]<='
                now=now*10;
                now+=(function_map[i+j]-'0');
            }
            if(val == 0 )val = 1;
            //printf("val:%d \n",val);
            re[k][now]+=val*bo;
            //printf("now:%d \n",now);
            val = 0;
            bo = 1;
            i=i+j-1;
        }
        else if(function_map[i]=='-'){
            bo = -1;
        }
        else if(function_map[i]>='0'&&function_map[i]<='9'){
            for(j=0;function_map[i+j]>='0'&&function_map[i+j]<='
                val=val*10;
                val+=(function_map[i+j]-'0');
            }
            i=i+j-1;
        }
    }
    constant = val*bo;
}    You, 8 hours ago • all

void itoa (int n, char s[])
{ ⋯
}

char* compute_and_combine(char* str1,char* str2, char op){ ⋯
}

void get_number(){ ⋯
}

char* derive_function(char var){ ⋯
}
```

CFG設計，其中使運算符號可以連續使用(ex. 多項式 + 多項式 + ...)，且進行優先序的設計，微分會先運算

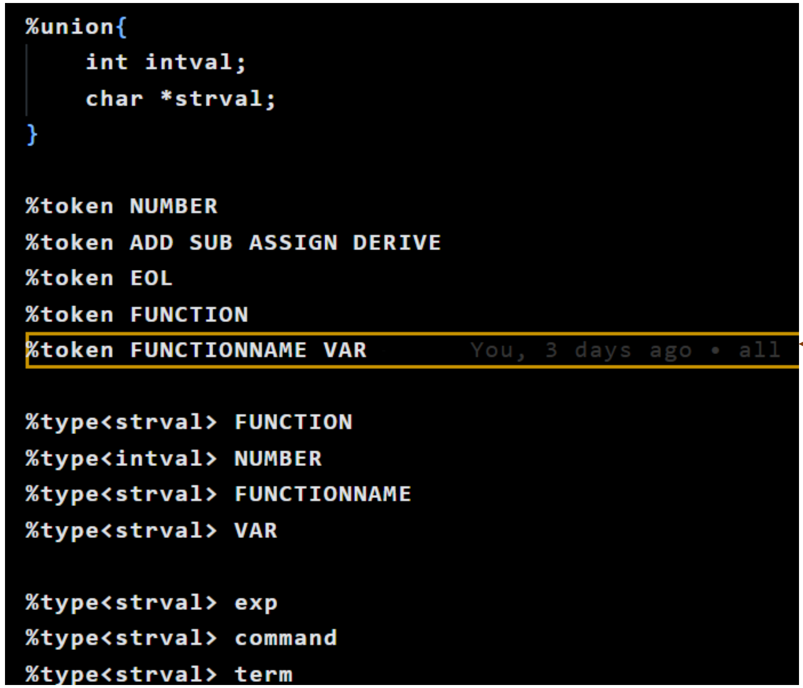


```
%%
commandlist:
|commandlist command EOL

command:
| FUNCTIONNAME ASSIGN exp {
    strcpy(function_map[(int)($1[0]-'A')],$3);
} // F = ...

| VAR ASSIGN NUMBER {
    if($1[0] == 'x'){
        x = $3;
    }
    else if($1[0] == 'y'){
        y = $3;
    }
    else if($1[0] == 'z'){
        z = $3;
    }
    else{
        printf("error\n");
    }
    int i;
    for(i=0;i<26;i++){
        if(strlen(function_map[i]) != 0){
            printf("origin %c: %s\n", i+'A',function_map[i]);
            string_to_array(function_map[i]);
            get_number();
            printf("answer %c: %s \n\n",i+'A',return_string);
        }
    }
    printf("------------------\n");
    if(x==2e9){
        printf("|x: | undefined   |\n");
    }

    if(x!=2e9){
        printf("|x: |}      |\n", x);
    }
    printf("------------------\n");
    if(y == 2e9){
        printf("|y: | undefined   |\n");
    }

    if(y!=2e9){
        printf("|y: |}      |\n", y);
    }
    printf("------------------\n");
    if(z == 2e9){
        printf("|z: | undefined   |\n");
    }

    if(z!=2e9){
        printf("|z: |}      |\n", z);
    }
    printf("------------------\n");
}



exp: term
|exp ADD term{
    // printf("%s + %s =", $1 , $3) ;
    $$ = compute_and_combine($1, $3, '+');
    // printf("output: %s\n", $$);
}

|exp SUB term   {
    // printf("%s + %s =", $1 , $3) ;
    $$ = compute_and_combine($1, $3, '-');
    // printf("output: %s\n", $$);
}

|exp ADD term   {/*$$=$1+function_map[$3];*/
    if(strlen(function_map[(int)($3[0]-'A')])==0){
        printf("%s is undefined\n", $3);
    }
    else{
        $$ = compute_and_combine($1, function_map[(int)($3[0]-'A')], '+');
    }
}
|exp SUB term   {/*$$=$1-function_map[$3];*/
    if(strlen(function_map[(int)($3[0]-'A')])==0){
        printf("%s is undefined\n", $3);
    }
    else{
        $$ = compute_and_combine($1, function_map[(int)($3[0]-'A')], '-');
    }
}


term: term DERIVE VAR {
    string_to_array($1);
    $$ = derive_function($3[0]);
}
|FUNCTION{strcpy($$,$1);}
|FUNCTIONNAME{
    if(strlen(function_map[(int)($1[0]-'A')])==0){
        printf("%s is undefined\n", $1);
    }else{
        $$ = function_map[(int)($1[0]-'A')];
    }
}

%%
```



```
%{
    #include <stdio.h>
    #include <string.h>
    #include <math.h>
    char function_map[30][200] = {0}; // copy:   strcpy(function_map["變數
    "-'A'], str)
    int x = 2e9, y = 2e9, z = 2e9;  //(x==2e9)?not define : x

    int re[3][200];
    int constant = 0;
    char return_string[1000];
```

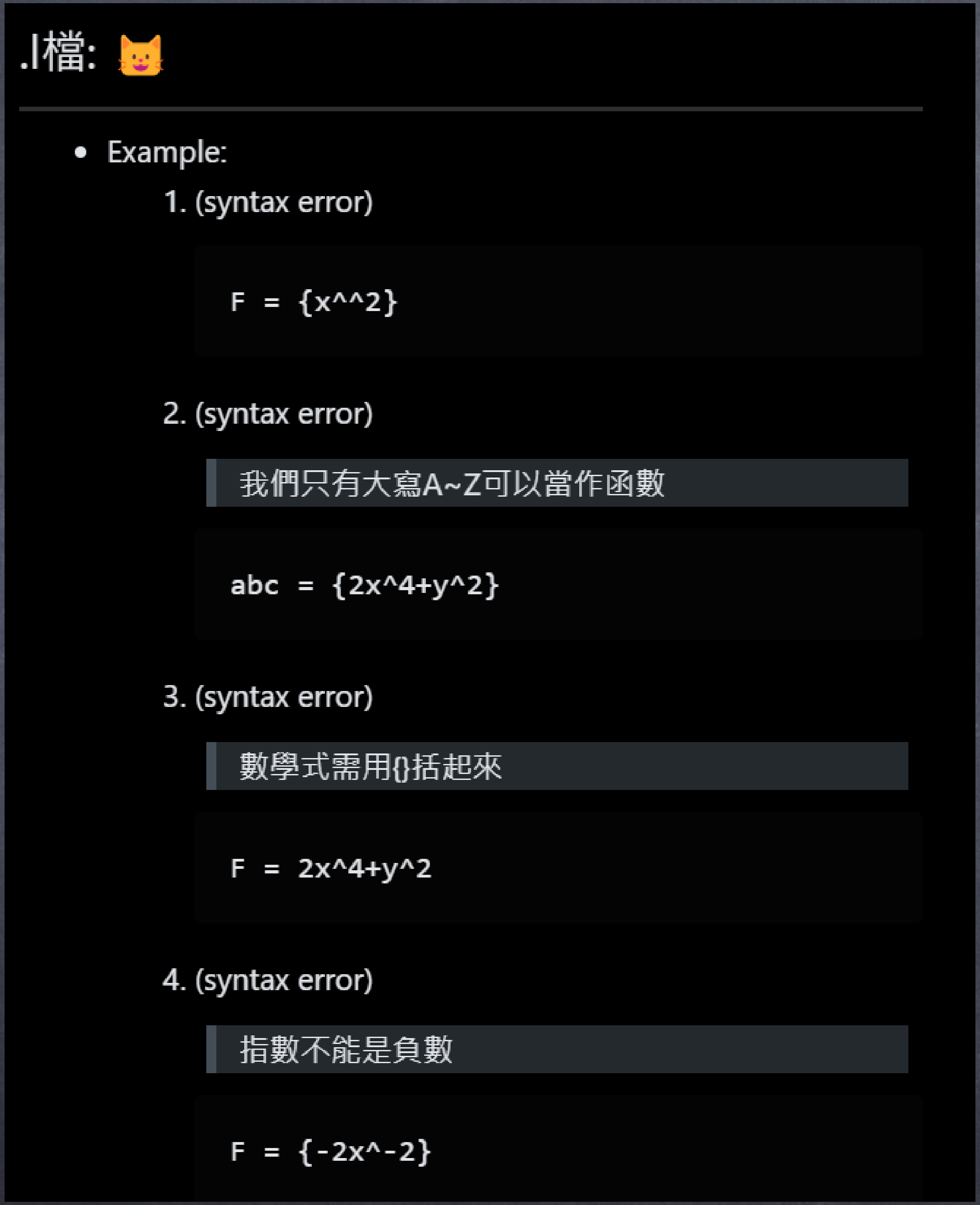
運用資料結構進行全域宣告，儲存使用者的變數資訊



```
%union{
    int intval;
    char *strval;
}

%token NUMBER
%token ADD SUB ASSIGN DERIVE
%token EOL
%token FUNCTION
%token FUNCTIONNAME VAR          You, 3 days ago • all

%type<strval> FUNCTION
%type<intval> NUMBER
%type<strval> FUNCTIONNAME
%type<strval> VAR

%type<strval> exp
%type<strval> command
%type<strval> term
```

查閱Bison文件，使用union，讓.l檔傳遞過來的資訊可以為char*與int

Demo & 測試結果



.l檔: 🐱

- Example:
    1. (syntax error)

       ```
       F = {x^^2}
       ```

    2. (syntax error)

       > 我們只有大寫A~Z可以當作函數

       ```
       abc = {2x^4+y^2}
       ```

    3. (syntax error)

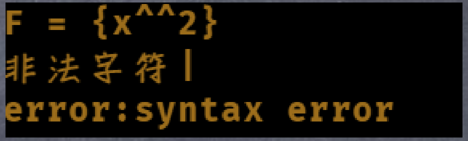
       > 數學式需用{}括起來

       ```
       F = 2x^4+y^2
       ```

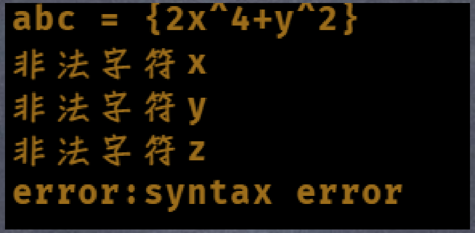
    4. (syntax error)

       > 指數不能是負數

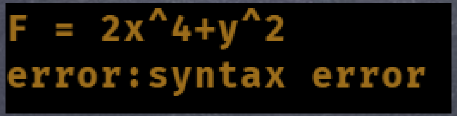
       ```
       F = {-2x^-2}
       ```



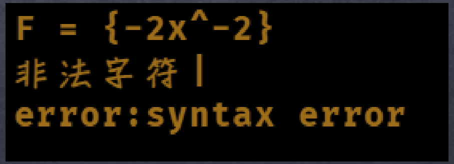
```
F = {x^^2}
非法字符|
error:syntax error
```



```
abc = {2x^4+y^2}
非法字符x
非法字符y
非法字符z
error:syntax error
```



```
F = 2x^4+y^2
error:syntax error
```



```
F = {-2x^-2}
非法字符|
error:syntax error
```

.y檔: 🐶

- 規則: 輸入任意多項式，可連續做多項式運算，直到對變數(x,y,z)賦值，才會輸出結果。
- Example: 0. 全部token都存在，但因為排列錯誤，CFG判斷為error `F = {3x^3+2x} +`

1. variable沒有全部賦值，可得部分解

```
F = {9z^5+x^5+x^4+1}
z = -1
x = 1
```

2. 可以連續做函式加法/減法/偏微分

> 可使用函式做運算

- 加法:

```
F = {2x^5+3x^4+5y^2} + {x^5+x^2+z^4} + {24}
G = F + {3x^2}
x = 1
x = 0
y = -10
z = 20
```

- 減法:

```
F = {2x^5+3x^4+5y^2} - {x^5+x^2+z^4}
G = {3x^2} - F
x = 1
x = 0
y = -10
z = 2
```

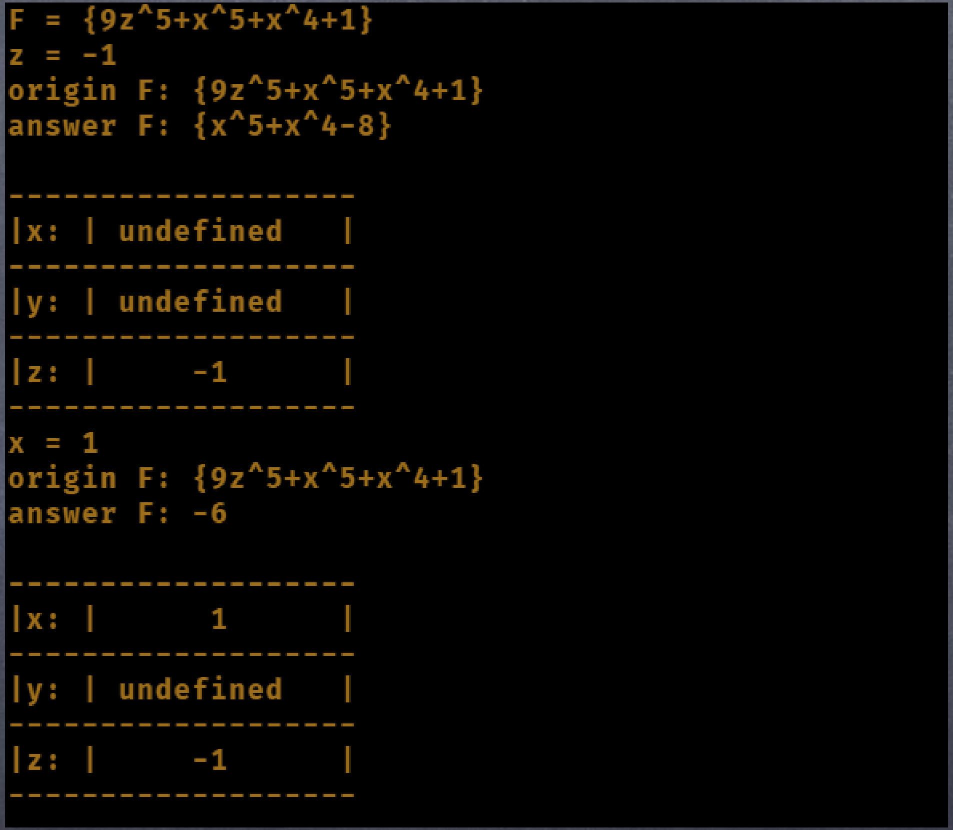
- 偏微分:

```
F = {2x^100-3x^5+9y^5+5z^1+6}
G = F |d x
H = F |d y
I = F |d z
K = F|dx |dy |dz
x = 1
```

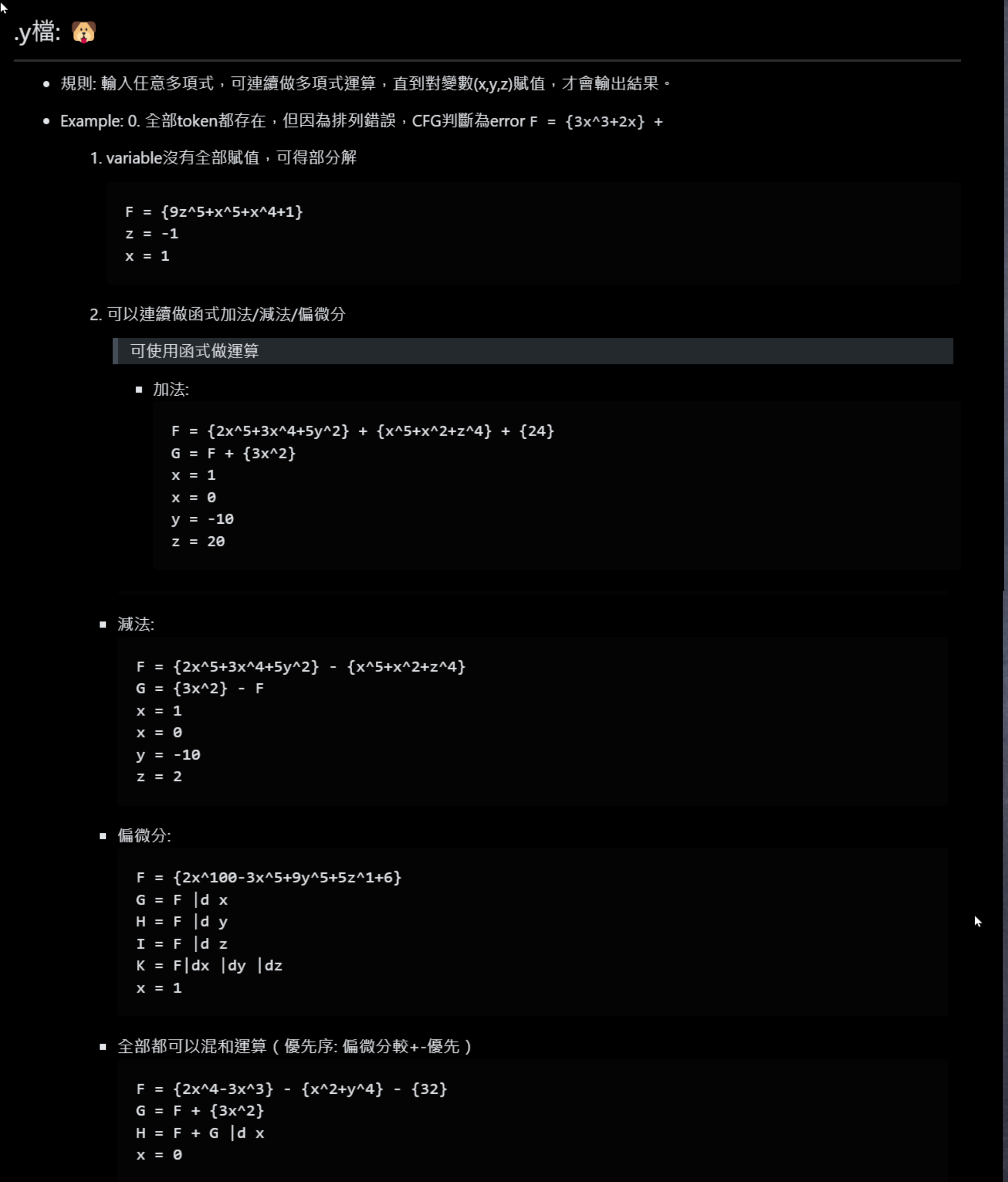
- 全部都可以混和運算 (優先序: 偏微分較+-優先)

```
F = {2x^4-3x^3} - {x^2+y^4} - {32}
G = F + {3x^2}
H = F + G |d x
x = 0
```

1.



```
F = {9z^5+x^5+x^4+1}
z = -1
origin F: {9z^5+x^5+x^4+1}
answer F: {x^5+x^4-8}

-------------------
|x: | undefined   |
-------------------
|y: | undefined   |
-------------------
|z: |     -1      |
-------------------
x = 1
origin F: {9z^5+x^5+x^4+1}
answer F: -6

-------------------
|x: |      1      |
-------------------
|y: | undefined   |
-------------------
|z: |     -1      |
-------------------
```



.y檔: 🐶

- 規則: 輸入任意多項式，可連續做多項式運算，直到對變數(x,y,z)賦值，才會輸出結果。
- Example: 0. 全部token都存在，但因為排列錯誤，CFG判斷為error `F = {3x^3+2x} +`

  1. variable沒有全部賦值，可得部分解

```
F = {9z^5+x^5+x^4+1}
z = -1
x = 1
```

  2. 可以連續做函式加法/減法/偏微分

> 可使用函式做運算

- 加法:

```
F = {2x^5+3x^4+5y^2} + {x^5+x^2+z^4} + {24}
G = F + {3x^2}
x = 1
x = 0
y = -10
z = 20
```

- 減法:

```
F = {2x^5+3x^4+5y^2} - {x^5+x^2+z^4}
G = {3x^2} - F
x = 1
x = 0
y = -10
z = 2
```

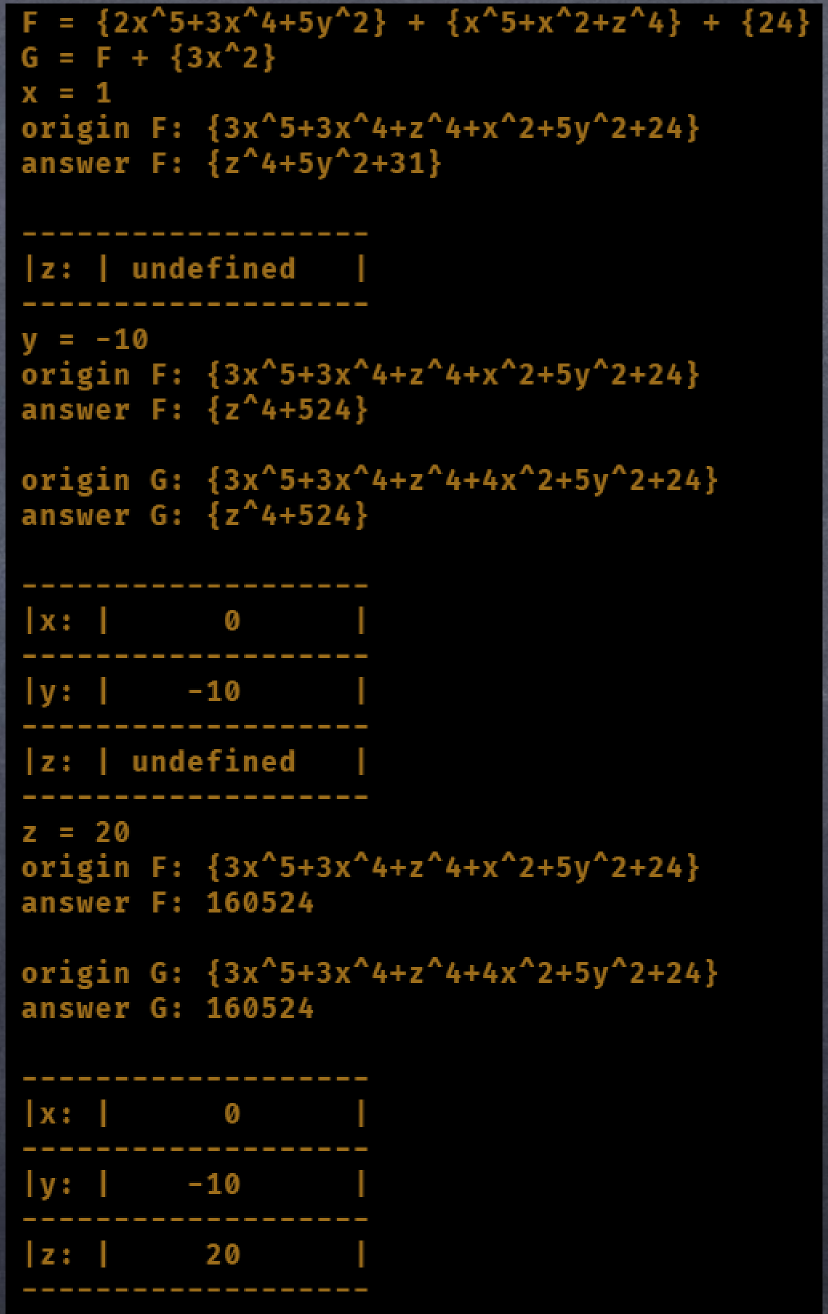
- 偏微分:

```
F = {2x^100-3x^5+9y^5+5z^1+6}
G = F |d x
H = F |d y
I = F |d z
K = F|dx |dy |dz
x = 1
```

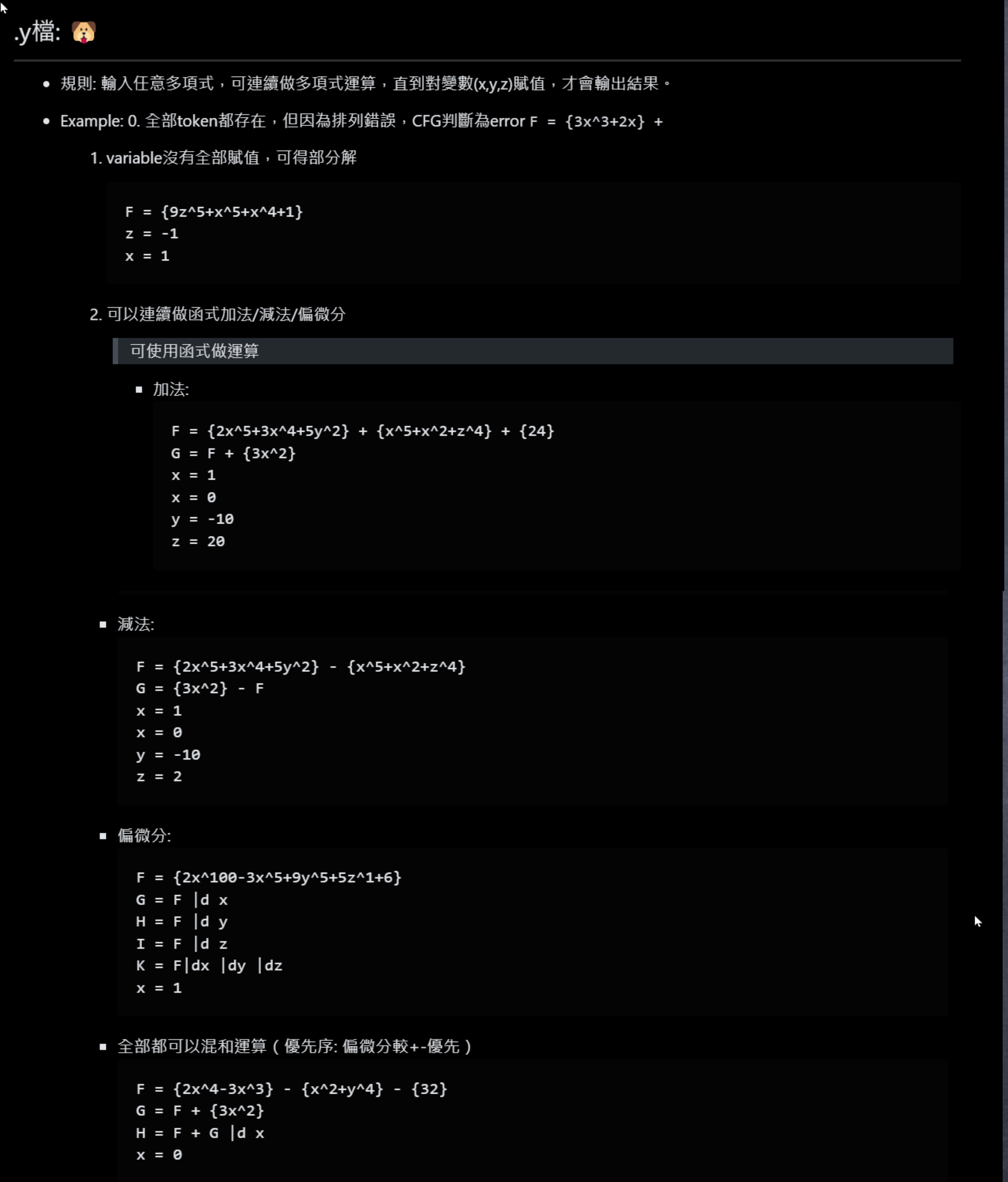
- 全部都可以混和運算（優先序: 偏微分較+-優先）

```
F = {2x^4-3x^3} - {x^2+y^4} - {32}
G = F + {3x^2}
H = F + G |d x
x = 0
```

2. <加法>



```
F = {2x^5+3x^4+5y^2} + {x^5+x^2+z^4} + {24}
G = F + {3x^2}
x = 1
origin F: {3x^5+3x^4+z^4+x^2+5y^2+24}
answer F: {z^4+5y^2+31}

-------------------
|z: | undefined    |
-------------------
y = -10
origin F: {3x^5+3x^4+z^4+x^2+5y^2+24}
answer F: {z^4+524}

origin G: {3x^5+3x^4+z^4+4x^2+5y^2+24}
answer G: {z^4+524}


-------------------
|x: |      0       |
-------------------
|y: |    -10       |
-------------------
|z: | undefined    |
-------------------
z = 20
origin F: {3x^5+3x^4+z^4+x^2+5y^2+24}
answer F: 160524

origin G: {3x^5+3x^4+z^4+4x^2+5y^2+24}
answer G: 160524


-------------------
|x: |      0       |
-------------------
|y: |    -10       |
-------------------
|z: |     20       |
-------------------
```



.y檔: 🐶

- 規則: 輸入任意多項式，可連續做多項式運算，直到對變數(x,y,z)賦值，才會輸出結果。
- Example: 0. 全部token都存在，但因為排列錯誤，CFG判斷為error `F = {3x^3+2x} +`
    1. variable沒有全部賦值，可得部分解

```
F = {9z^5+x^5+x^4+1}
z = -1
x = 1
```

2. 可以連續做函式加法/減法/偏微分

> 可使用函式做運算

- 加法:

```
F = {2x^5+3x^4+5y^2} + {x^5+x^2+z^4} + {24}
G = F + {3x^2}
x = 1
x = 0
y = -10
z = 20
```

- 減法:

```
F = {2x^5+3x^4+5y^2} - {x^5+x^2+z^4}
G = {3x^2} - F
x = 1
x = 0
y = -10
z = 2
```

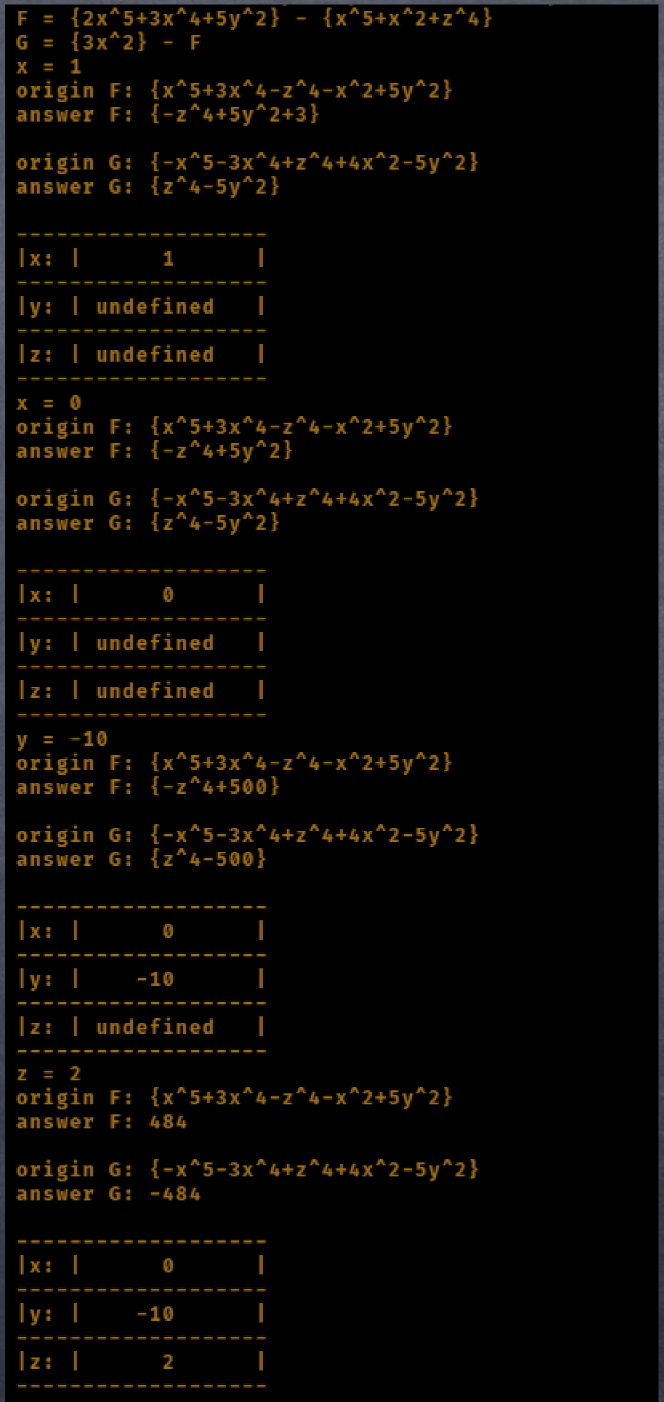
- 偏微分:

```
F = {2x^100-3x^5+9y^5+5z^1+6}
G = F |d x
H = F |d y
I = F |d z
K = F|dx |dy |dz
x = 1
```

- 全部都可以混和運算（優先序: 偏微分較+-優先）

```
F = {2x^4-3x^3} - {x^2+y^4} - {32}
G = F + {3x^2}
H = F + G |d x
x = 0
```

2. <減法>



```
F = {2x^5+3x^4+5y^2} - {x^5+x^2+z^4}
G = {3x^2} - F
x = 1
origin F: {x^5+3x^4-z^4-x^2+5y^2}
answer F: {-z^4+5y^2+3}

origin G: {-x^5-3x^4+z^4+4x^2-5y^2}
answer G: {z^4-5y^2}

-------------------
|x: |      1       |
-------------------
|y: | undefined    |
-------------------
|z: | undefined    |
-------------------
x = 0
origin F: {x^5+3x^4-z^4-x^2+5y^2}
answer F: {-z^4+5y^2}

origin G: {-x^5-3x^4+z^4+4x^2-5y^2}
answer G: {z^4-5y^2}

-------------------
|x: |      0       |
-------------------
|y: | undefined    |
-------------------
|z: | undefined    |
-------------------
y = -10
origin F: {x^5+3x^4-z^4-x^2+5y^2}
answer F: {-z^4+500}

origin G: {-x^5-3x^4+z^4+4x^2-5y^2}
answer G: {z^4-500}

-------------------
|x: |      0       |
-------------------
|y: |    -10       |
-------------------
|z: | undefined    |
-------------------
z = 2
origin F: {x^5+3x^4-z^4-x^2+5y^2}
answer F: 484

origin G: {-x^5-3x^4+z^4+4x^2-5y^2}
answer G: -484

-------------------
|x: |      0       |
-------------------
|y: |    -10       |
-------------------
|z: |      2       |
-------------------
```

.y檔: 🐶

- 規則: 輸入任意多項式，可連續做多項式運算，直到對變數(x,y,z)賦值，才會輸出結果。
- Example: 0. 全部token都存在，但因為排列錯誤，CFG判斷為error `F = {3x^3+2x} +`

  1. variable沒有全部賦值，可得部分解

     ```
     F = {9z^5+x^5+x^4+1}
     z = -1
     x = 1
     ```

  2. 可以連續做函式加法/減法/偏微分

     > 可使用函式做運算

     - 加法:

       ```
       F = {2x^5+3x^4+5y^2} + {x^5+x^2+z^4} + {24}
       G = F + {3x^2}
       x = 1
       x = 0
       y = -10
       z = 20
       ```

     - 減法:

       ```
       F = {2x^5+3x^4+5y^2} - {x^5+x^2+z^4}
       G = {3x^2} - F
       x = 1
       x = 0
       y = -10
       z = 2
       ```

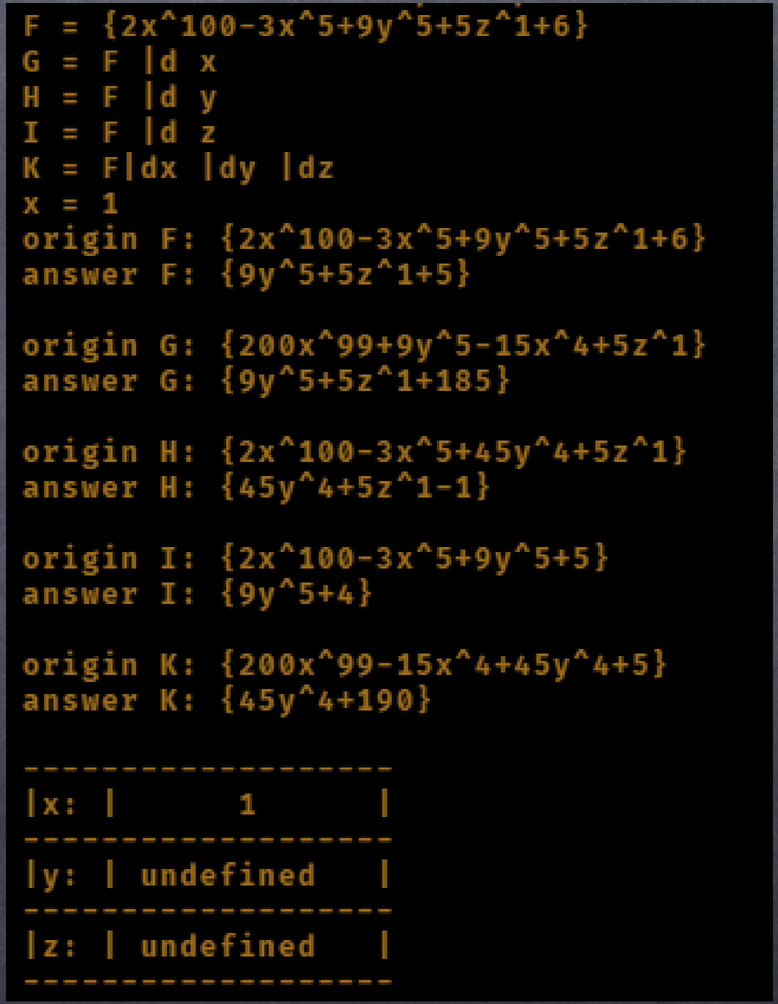
     - 偏微分:

       ```
       F = {2x^100-3x^5+9y^5+5z^1+6}
       G = F |d x
       H = F |d y
       I = F |d z
       K = F|dx |dy |dz
       x = 1
       ```

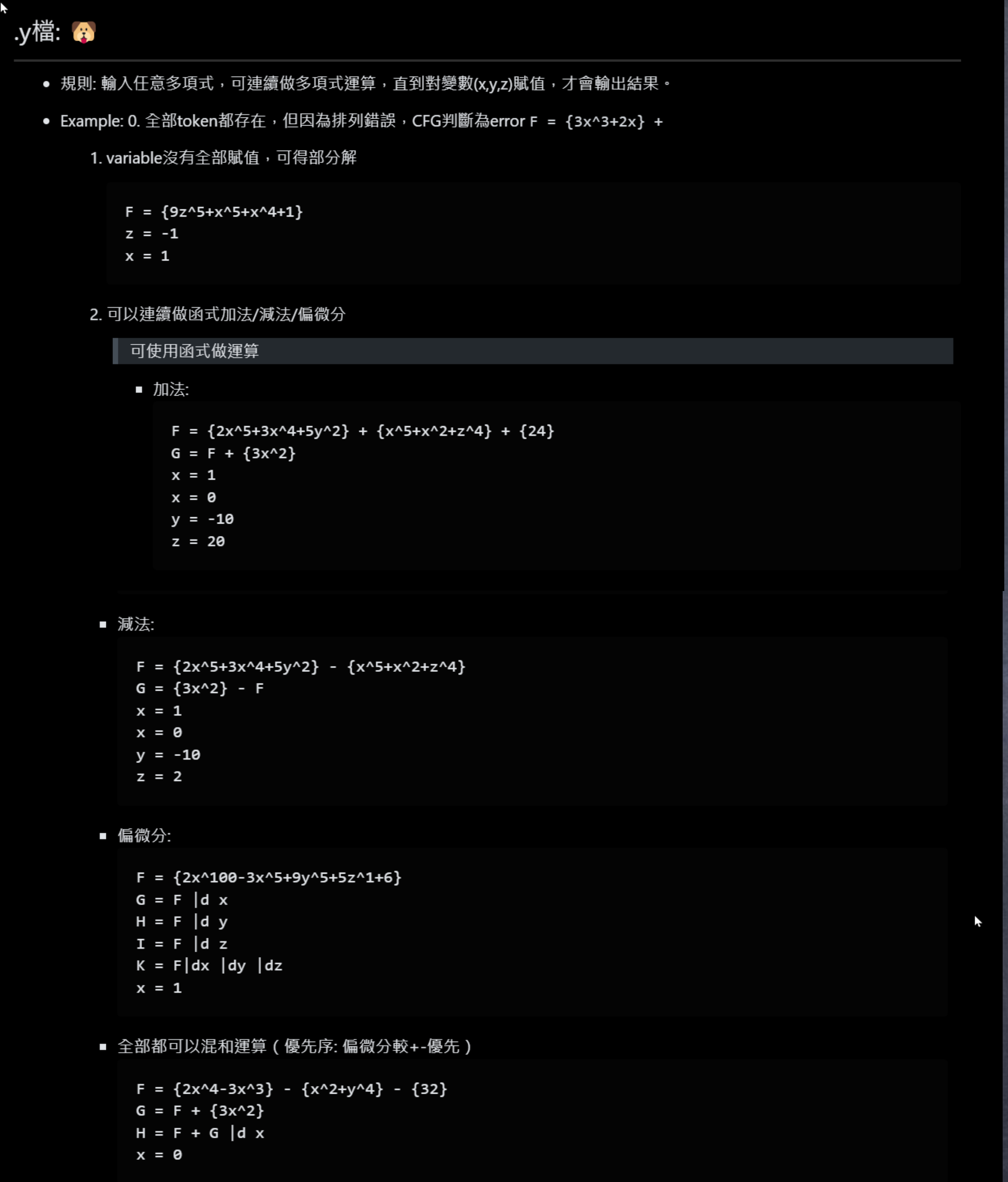
     - 全部都可以混和運算（優先序: 偏微分較+-優先）

       ```
       F = {2x^4-3x^3} - {x^2+y^4} - {32}
       G = F + {3x^2}
       H = F + G |d x
       x = 0
       ```

2. <偏微分>



```
F = {2x^100-3x^5+9y^5+5z^1+6}
G = F |d x
H = F |d y
I = F |d z
K = F|dx |dy |dz
x = 1
origin F: {2x^100-3x^5+9y^5+5z^1+6}
answer F: {9y^5+5z^1+5}

origin G: {200x^99+9y^5-15x^4+5z^1}
answer G: {9y^5+5z^1+185}

origin H: {2x^100-3x^5+45y^4+5z^1}
answer H: {45y^4+5z^1-1}

origin I: {2x^100-3x^5+9y^5+5}
answer I: {9y^5+4}

origin K: {200x^99-15x^4+45y^4+5}
answer K: {45y^4+190}

-------------------
|x: |       1      |
-------------------
|y: | undefined    |
-------------------
|z: | undefined    |
-------------------
```



.y檔: 🐶

- 規則: 輸入任意多項式，可連續做多項式運算，直到對變數(x,y,z)賦值，才會輸出結果。
- Example: 0. 全部token都存在，但因為排列錯誤，CFG判斷為error `F = {3x^3+2x} +`
  1. variable沒有全部賦值，可得部分解

```
F = {9z^5+x^5+x^4+1}
z = -1
x = 1
```

  2. 可以連續做函式加法/減法/偏微分

> 可使用函式做運算

- 加法:

```
F = {2x^5+3x^4+5y^2} + {x^5+x^2+z^4} + {24}
G = F + {3x^2}
x = 1
x = 0
y = -10
z = 20
```

- 減法:

```
F = {2x^5+3x^4+5y^2} - {x^5+x^2+z^4}
G = {3x^2} - F
x = 1
x = 0
y = -10
z = 2
```

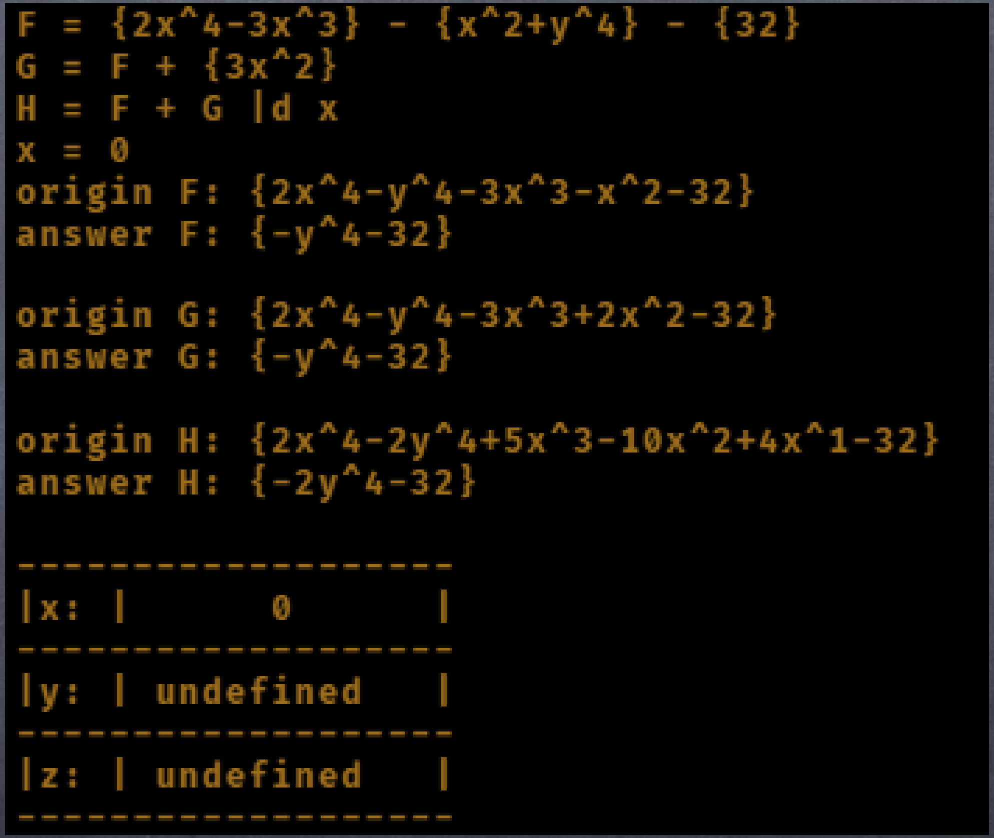
- 偏微分:

```
F = {2x^100-3x^5+9y^5+5z^1+6}
G = F |d x
H = F |d y
I = F |d z
K = F|dx |dy |dz
x = 1
```

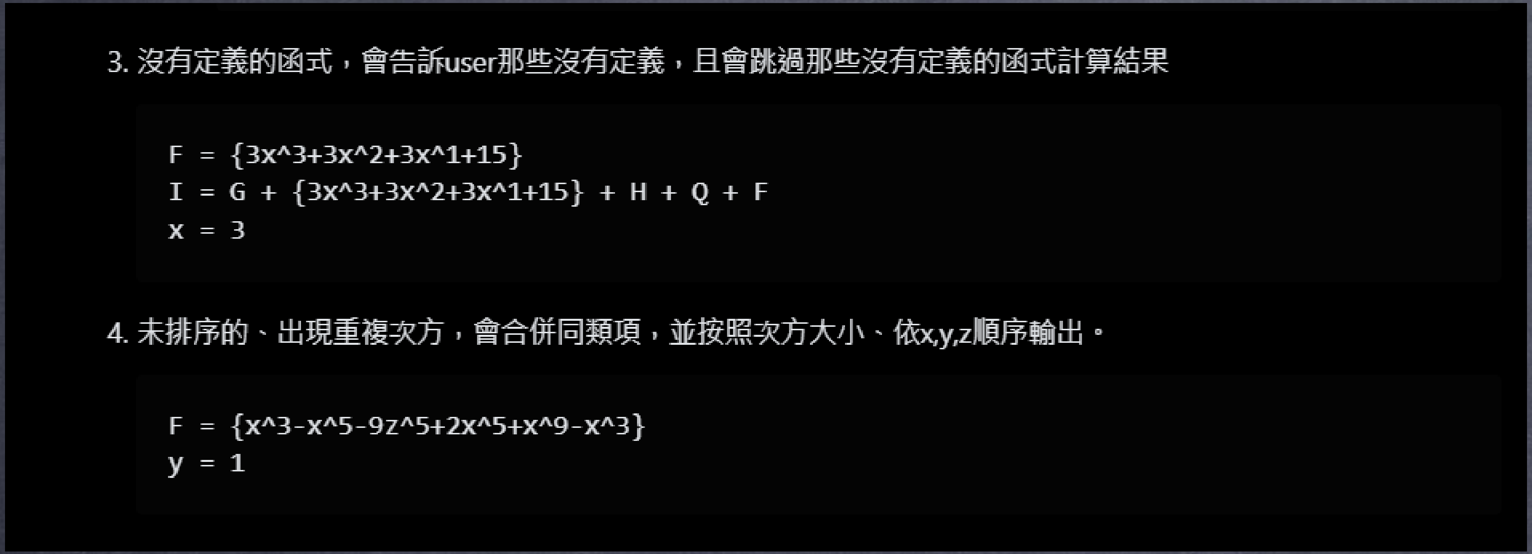
- 全部都可以混和運算 (優先序: 偏微分較+-優先)

```
F = {2x^4-3x^3} - {x^2+y^4} - {32}
G = F + {3x^2}
H = F + G |d x
x = 0
```

2. <全部都可以混合運算>



```
F = {2x^4-3x^3} - {x^2+y^4} - {32}
G = F + {3x^2}
H = F + G |d x
x = 0
origin F: {2x^4-y^4-3x^3-x^2-32}
answer F: {-y^4-32}

origin G: {2x^4-y^4-3x^3+2x^2-32}
answer G: {-y^4-32}

origin H: {2x^4-2y^4+5x^3-10x^2+4x^1-32}
answer H: {-2y^4-32}

-------------------
|x: |      0      |
-------------------
|y: | undefined   |
-------------------
|z: | undefined   |
-------------------
```



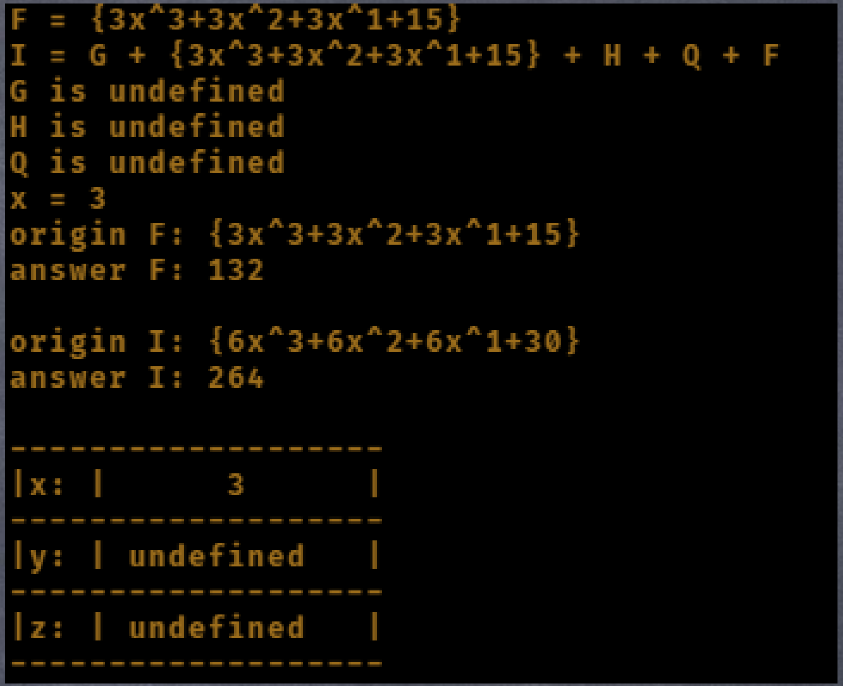
3. 沒有定義的函式，會告訴user那些沒有定義，且會跳過那些沒有定義的函式計算結果

```
F = {3x^3+3x^2+3x^1+15}
I = G + {3x^3+3x^2+3x^1+15} + H + Q + F
x = 3
```

4. 未排序的、出現重複次方，會合併同類項，並按照次方大小、依x,y,z順序輸出。

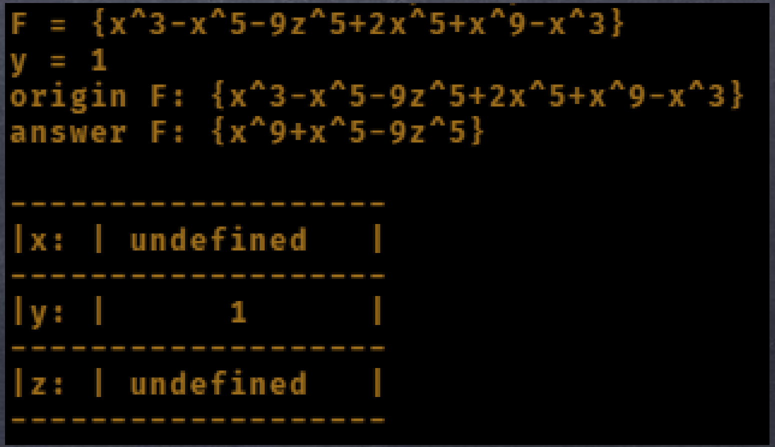
```
F = {x^3-x^5-9z^5+2x^5+x^9-x^3}
y = 1
```

3.



```
F = {3x^3+3x^2+3x^1+15}
I = G + {3x^3+3x^2+3x^1+15} + H + Q + F
G is undefined
H is undefined
Q is undefined
x = 3
origin F: {3x^3+3x^2+3x^1+15}
answer F: 132

origin I: {6x^3+6x^2+6x^1+30}
answer I: 264

-------------------
|x: |      3      |
-------------------
|y: | undefined   |
-------------------
|z: | undefined   |
-------------------
```

4.



```
F = {x^3-x^5-9z^5+2x^5+x^9-x^3}
y = 1
origin F: {x^3-x^5-9z^5+2x^5+x^9-x^3}
answer F: {x^9+x^5-9z^5}

-------------------
|x: | undefined   |
-------------------
|y: |      1      |
-------------------
|z: | undefined   |
-------------------
```

專案遇到的問題

1. Bison語法不熟悉

   (1) 在改變型態和union使用的地方研究許久

   (2) 對於如何在Bison上定義函式也花了一點時間

2. 分段被return的問題

   設計多項式的正規式時，

   一開始設計的會出現分段被return的問題，

   因為在讀取token時只要先符合某一正規式則會被return

   因此我們利用”{“和”}”將多項式包起來，讓我們在解析token時可以整段讀取

3. 一開始設計CFG時，不知道如何處理優先序的問題

   本來以為微分與+-列在同一個產生式只要微分在上面就可以了，

   但後來發現要額外列新的一條產生式且要在此之下，

   bottom-up系列方法建tree時才會先處理

4. 研究哪種資料結構比較適合處理多項式

   最後採用二維陣列來處理多項式