

电子科技大学信息与软件工程学院

实 验 报 告

学 号 2018091618008

姓 名 袁昊男

(实验) 课程名称 信息安全数学基础

理论教师 熊虎

实验教师 熊虎

电子科技大学

实验报告

学生姓名：袁昊男 学号：2018091618008 指导教师：熊虎

实验地点：信软楼 303 实验时间：2019.12.15

一、实验名称：有限域上运算的实现

二、实验学时：2 学时

三、实验目的：

- 1、加深对有限域结构的理解，实现常用的有限域上元素的基本运算；
- 2、编程实现 $GF(2^8)$ 域中元素的运算，提供使用说明手册，可执行代码、源代码及测试用例（包括测试用例手册、可执行代码和源代码）。

四、实验原理：

- 1、算法 7.5.1 在 F_{p^n} 中计算乘法逆元。

输入：非零多项式 $g(\alpha) \in F_{p^n}$ （ F_{p^n} 中的元素以 $f(x)$ 的根 α 的次数小于 n

的多项式形式表示，其中 $f(x) \in F_p[X]$ 是 \mathbb{Z}_p 上的次数为 n 的不可约多项式）。

输出： $g^{-1}(\alpha) \in F_{p^n}$ 。

- (1) 利用适用于多项式的扩展的欧几里得算法（算法 6.2.2）得出两个多项式 $s(\alpha), t(\alpha) \in F_p(\alpha)$ ，使得 $s(\alpha)g(\alpha) + t(\alpha)f(\alpha) = 1$ 。

- (2) 返回 $(s(\alpha))$ 。
-

- 2、 F_{p^n} 中的幂运算可以用重复平方乘算法有效地计算得出。

算法 7.5.2 适用于 F_{p^n} 中幂运算的重复平方乘算法。

输入： $g(\alpha) \in F_{p^n}$ ，整数 $0 \leq k \leq p^n - 1$ ，其二进制表示为 $k = \sum_{i=0}^t k_i 2^i$ 。

（ F_{p^n} 中的元素以 $f(x)$ 的根 α 的次数小于 n 的多项式形式表示，

其中 $f(x) \in F_p[X]$ 是 \mathbb{Z}_p 上的次数为 n 的不可约多项式）。

输出: $g(\alpha)^k$ 。

(1) 令 $s(\alpha) \leftarrow 1$, 如果 $k=0$, 返回 $(s(\alpha))$ 。

(2) 令 $G(\alpha) \leftarrow g(\alpha)$ 。

(3) 如果 $k_0=1$, 则令 $s(\alpha) \leftarrow g(\alpha)$ 。

(4) 对 i 从 1 到 t 作

(4.1) 令 $G(\alpha) \leftarrow G^2(\alpha) \bmod f(\alpha)$;

(4.2) 如果 $k_i=1$, 则令 $s(\alpha) \leftarrow G(\alpha) \cdot s(\alpha) \bmod f(\alpha)$ 。

(5) 返回 $(s(\alpha))$ 。

3、GF 域中的多项式加法与乘法一般方法: 首先将域中的元素表示为其对应的二进制码。如在 $\text{GF}(2^8)$ 中, $f(x)=x^7+x^6+x^5+x+1$ 对应的二进制码是 11100011; 反之 10101011 对应的多项式为 $g(x)=x^7+x^5+x^3+x+1$ 。在这样的表示情况下实现一般加法和乘法就十分简单:

(1) 加法: 即对应分量的简单相加。如 $f(x)=x^3+x+1$, $g(x)=x^3+1$ 相加, 即 $(1011)+(1001)=(0010)$, 对应的结果多项式为 x 。

(2) 乘法: 如 $f(x)=x^4+x^3+x+1$, $g(x)=x^3+x^2+x$ 相乘, 即 $(11011) \times (1110)$, 我们可以把这个计算拆开为 $(11011) \times (1000) + (11011) \times (100) + (11011) \times (10) + (11011) \times (0)$, 而且我们知道对于每一个乘式, 都是对左数进行移位, 即: $(11011) \ll 3 + (11011) \ll 2 + (11011) \ll 1$, 因此最后的结果为 (10000010) , 对应的结果多项式为 x^7+x 。

4、构造指数对数表:

取域 F_2 上的 8 次不可约多项式 $f(x)=x^8+x^6+x^5+x+1$, α 是 $f(x)$ 的一个根。因此有限域 $\text{GF}(2^8)$ 可以表示为 α 的所有 F_2 次数小于 8 的多项式集合, 即

$$F_{2^8} = \{a_7\alpha^7 + a_6\alpha^6 + a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 \mid a_i \in \{0,1\}\}$$

定义一个由 $a_7a_6a_5a_4a_3a_2a_1a_0$ 组成的字节 a 可表示为系数 $\{0,1\}$ 的二进制多项式:

$$a_7\alpha^7 + a_6\alpha^6 + a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$$

还可以将每个字节表示为一个十六进制数, 即每 4 比特表示一个十六进制数, 代表较高位的 4 比特的符号仍在左边。例如, 01101011 可表示为 6B。同样也可以用 0~255 这 256 个十进制整数来表示域 $\text{GF}(2^8)$ 中的元素。它们之间的运算为 $\text{GF}(2^8)$ 中的运算, 其加法定义为二进制多项式的加法, 且其系数模 2, 其乘法定义为多项式的乘积模一个次数为 8 的不可约多项式 $f(\alpha)$ 。通过计算机实验可以验证元素 “02” 是域 $\text{GF}(2^8)$ 中的一个本原元。

将域 $\text{GF}(2^8)$ 中的元素用 0~255 这 256 个十进制整数来表示, 指数对数表可以通过如下步骤来构建:

- (1) 将元素 ‘02’ 表示成为 α ，依次计算 $\alpha^i \bmod(f(\alpha))$ ， $i = 0, 1, \dots, 254$ ，将所得结果转变为十进制数，设为 β_i ， $i = 0, 1, \dots, 254$ ；
- (2) 建表。第一行为 $0, 1, \dots, 254, 255$ ，第二行元素依次为 β_i ， $i = 0, 1, \dots, 254$ 。由于 $\alpha^0 \equiv \alpha^{255} \bmod(f(\alpha))$ ，约定第 2 行，第 255 列元素为 0，如下表所示：

0	1	2	3	...	253	254	255
1	2	4	8	...	233	177	0

- (3) 按所建表的第二行元素的大小进行重排列，如下表所示：

255	0	1	197	...	72	230	104
0	1	2	3	...	253	254	255

- (4) 将(3)中表的第一行放在(2)中表的第三行，即：

序号	0	1	2	3	...	253	254	255
$(02)^i$	1	2	4	8	...	233	177	0
$\log_{(02)^i}$	255	0	1	197	...	72	230	104

建立上述指数对数表后，通过查表很容易求出两个元素的乘积。又由于对于 $i = 0, 1, \dots, 254$ 均有 $(\alpha^i)^{-1} \equiv \alpha^{255-i} \bmod(f(\alpha))$ ，所以通过查表也很容易求出元素的逆元。

五、 实验内容：

- 1、实现 2^8 域上元素的多项式基表示，实现模多项式的乘法运算和求逆运算；
- 2、实现 2^8 域上元素乘法运算和逆元运算；
- 3、构造指数对数表，从而通过查表实现 2^8 域上元素乘法运算和逆元运算；
- 4、不可约多项式为 $f(x) = x^8 + x^6 + x^5 + x + 1$ ；
- 5、本原元为 “02”；
- 6、按照标准实验报告整理实验内容。

六、 实验器材（设备、元器件）：

电脑一台。

七、 实验步骤：

- 1、学习实验原理，尤其是指数对数表的构造方法，理解算法实现的过程；
- 2、分析题目需求，设计数据结构；
- 3、编码实现，并按测试用例进行输入输出测试；

4、分析实验结果，撰写实验报告。

八、实验结果与分析（含重要数据结果分析或核心代码流程分析）

1、实现 2^8 域上元素的多项式基表示

(1) 代码

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5. int a[8] = { 0 };
6.
7. void AddOne(int a[])
8. {
9.     int i = 0;
10.    for (i = 0; i < 8; i++)
11.    {
12.        if (a[i] == 0)
13.        {
14.            a[i] = 1;
15.            break;
16.        }
17.        else if (a[i] == 1)
18.        {
19.            a[i] = 0;
20.            while (i != 8)
21.            {
22.                if (a[i + 1] == 0)
23.                {
24.                    a[i + 1] = 1;
25.                    break;
26.                }
27.                else if (a[i + 1] == 1)
28.                {
29.                    a[i + 1] = 0;
30.                    i++;
31.                }
32.            }
33.            break;
34.        }
35.    }
36.}
37.
38.int main()
39.{
40.    int i, j;
41.    for (i = 0; i < 256; i++)
42.    {
43.        for (j = 7; j >= 0; j--)
44.        {
45.            if (a[j] != 0)
46.                printf("x^%d ", j);
47.        }
48.        if (i == 0)
49.            printf("0");
50.        printf("\n");
```

```

51.     AddOne(a);
52. }
53.     system("pause");
54.     return 0;
55. }

```

说明：由实验原理，可以定义一个由 $a_7a_6a_5a_4a_3a_2a_1a_0$ 组成的字节 a 可表示为系数为{0,1}的二进制多项式：

$$a_7\alpha^7 + a_6\alpha^6 + a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$$

因此，以 int 型 8 位的全局数组 a 来保存多项式系数，每输出一个多项式， a 按照二进制加法增加 1。

(2) 运行截图

8 位二进制代码 $a_7a_6a_5a_4a_3a_2a_1a_0$ 可以表示从“0”到“255”对应的 256 个 $GF(2^8)$ 域上的多项式。

2、模多项式的乘法运算和求逆运算

(1) 代码

```

1. # 求最高幂次数
2. def Times(value):
3.     v2str = '{:09b}'.format(value)
4.     for i in range(9):
5.         if int(v2str[i]):
6.             return 9-i
7.

```

```

8. # 模 2 除法: m 为被除数。b 为除数, q 为商, r 为余数
9. def Mod2_Div(f, g):
10.     n = Times(f)
11.     m = Times(g)
12.     if n < m:
13.         return [0, f]
14.     deg = n - m
15.     f = f ^ (g << deg)
16.     [q, r] = Mod2_Div(f, g)
17.     return [(1 << deg)|q, r]
18.
19. # v3 = v1 - q3 * v2
20. def Calculate(v1, q3, v2):
21.     value = 0
22.     for i in range(32):
23.         if(q3 & (1<<i)):
24.             value = value ^ (v2<<i)
25.     return v1^value
26.
27. def Mul(a, b):
28.     value = 0
29.     for i in range(32):
30.         if(a & (1<<i)):
31.             value = value ^ (b<<i)
32.     return value
33.
34.
35. def Euclid(r1, r2, v1=1, v2=0, w1=0, w2=1):
36.     if r2 == 0 or r2 == 1: return w2
37.     q3, r3 = Mod2_Div(r1, r2) # q3(x)=r1(x)|r2(x), r2(x)
    )=r1(x) mod r2(x)
38.     v3 = Calculate(v1, q3, v2) # v3 = v1 - q3 * v2
39.     w3 = Calculate(w1, q3, w2) # w3 = w1 - q3 * w2
40.     return Euclid(r2, r3, v2, v3, w2, w3)
41.
42. def sym2int(sym):
43.     power = [sym[i+2] for i in range(len(sym)) if sym[i]
    == 'x']
44.     if '+1' in sym: power.append('0')
45.     data = 0
46.     for p in power:
47.         data = data | (1<<int(p))
48.     return data
49.
50. def int2sym(data):
51.     int2str = '{:09b}'.format(data)
52.     sym = ''
53.     for i in range(9):
54.         if int(int2str[i]) == 1:
55.             if 8-i: sym += '+x^%d'%(8-i)
56.             else: sym += '+1'
57.     return sym[1:]
58.
59. if __name__ == '__main__':
60.     print('-----多项式乘法测试-----')
61.     xstr1 = input('请输入多项式 f(x): ')
62.     xstr2 = input('请输入多项式 g(x): ')
63.

```

```

64.     print('f(x)·g(x) = ', int2sym(Mul(sym2int(xstr1), sy
      m2int(xstr2))))
65.
66.     print('\n')
67.     print('-----多项式求逆测试-----')
68.     xstr = input('请输入待求逆多项式 h(x): ')
69.     print('输入的多项式: ', xstr)
70.     print('不可约多项式多项式: ', int2sym(355))
71.     print('h(x)的乘法逆
      元: ', int2sym(Euclid(355, sym2int(xstr))))

```

说明：由实验原理，在 $GF(2^8)$ 多项式域上进行一般的乘法及求逆运算只需对多项式对应的二进制系数进行相应运算即可。其中主要函数是乘法运算 Mul 函数及求逆运算需要用到的 Euclid 函数。与实验三采用 C 语言完成扩展欧几里得的代码量对比，很明显 Python 在密码学相关算法的实现上优势明显，可读性更强。

(2) 测试用例

序号	输入 $(f(x), g(x), h(x))$	预期输出 $(f(x) \cdot g(x), h^{-1}(x))$	实际输出 $(f(x) \cdot g(x), h^{-1}(x))$
1	$f(x) = x^4 + x^3 + x + 1$ $g(x) = x^3 + x^2 + 1$ $h(x) = x^7 + x + 1$	$f(x) \cdot g(x) = x^7 + x^5 + x^3 + x^2 + x + 1$ $h^{-1}(x) = x^7 + x^6 + x^3 + 1$	$f(x) \cdot g(x) = x^7 + x^5 + x^3 + x^2 + x + 1$ $h^{-1}(x) = x^7 + x^6 + x^3 + 1$
2	$f(x) = x^3 + x^2 + x + 1$ $g(x) = x^4 + x^2 + 1$ $h(x) = x^5 + x^3 + 1$	$f(x) \cdot g(x) = x^7 + x^6 + x + 1$ $h^{-1}(x) = x^7 + x^5 + x^4 + x^2 + 1$	$f(x) \cdot g(x) = x^7 + x^6 + x + 1$ $h^{-1}(x) = x^7 + x^5 + x^4 + x^2 + 1$
3	$f(x) = x^2 + 1$ $g(x) = x^3 + x^2 + 1$ $h(x) = x^5 + x + 1$	$f(x) \cdot g(x) = x^5 + x^4 + x^3 + 1$ $h^{-1}(x) = x^4 + x^3 + x^2$	$f(x) \cdot g(x) = x^5 + x^4 + x^3 + 1$ $h^{-1}(x) = x^4 + x^3 + x^2$

(3) 运行截图

a) 测试用例 1

```

C:\袁昊男\学习\大二上\信息安全数学基础\实验\venv\Scripts\p
-----多项式乘法测试-----
请输入多项式f(x): x^4+x^3+x+1
请输入多项式g(x): x^3+x^2+1
f(x)·g(x) = x^7+x^5+x^3+x^2+x+1

-----多项式求逆测试-----
请输入待求逆多项式f(x): x^7+x+1
输入的多项式: x^7+x+1
不可约多项式多项式: x^8+x^6+x^5+x^1+1
f(x)的乘法逆元: x^7+x^6+x^3+1

Process finished with exit code 0

```


b) 测试用例 2

```
C:\袁昊男\学习\大二上\信息安全数学基础\实验\venv\Scripts\p
-----多项式乘法测试-----
请输入多项式f(x):  $x^3+x^2+x+1$ 
请输入多项式g(x):  $x^4+x^2+1$ 
f(x)·g(x) =  $x^7+x^6+x^1+1$ 

-----多项式求逆测试-----
请输入待求逆多项式f(x):  $x^5+x^3+1$ 
输入的多项式:  $x^5+x^3+1$ 
不可约多项式多项式:  $x^8+x^6+x^5+x^1+1$ 
f(x)的乘法逆元:  $x^7+x^5+x^4+x^2+1$ 

Process finished with exit code 0
```

c) 测试用例 3

```
C:\袁昊男\学习\大二上\信息安全数学基础\实验\venv\Scripts\p
-----多项式乘法测试-----
请输入多项式f(x):  $x^2+1$ 
请输入多项式g(x):  $x^3+x^2+1$ 
f(x)·g(x) =  $x^5+x^4+x^3+1$ 

-----多项式求逆测试-----
请输入待求逆多项式f(x):  $x^5+x+1$ 
输入的多项式:  $x^5+x+1$ 
不可约多项式多项式:  $x^8+x^6+x^5+x^1+1$ 
f(x)的乘法逆元:  $x^4+x^3+x^2$ 

Process finished with exit code 0
```

注：结果与手工计算验证正确。

3、构造指数对数表

(1) 代码

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5. int main()
6. {
7.     //正表
8.     int table[256];
9.     int i;
10.
11.     table[0] = 1; //  $\alpha^0$ 
12.     for (i = 1; i < 255; ++i) //生成元为  $\alpha$ 
13.     {
14.         table[i] = (table[i - 1] << 1) ;
15.
16.         //最高指数已经到了 8，需要模上不可约
17.         //  $f(x)=x^8+x^6+x^5+x+1$ 
18.         if (table[i] & 0x100)
19.             table[i] ^= 0x163; //  $f(x)=0x163$ 
20.     }
21. }
22. //反表
23. int arc_table[256];
```

```

24.   for (i = 0; i < 255; ++i)
25.       arc_table[table[i]] = i;
26.
27.   //逆元表
28.   int inverse_table[256];
29.
30.   for (i = 1; i < 256; ++i) //0 没有逆元，所以从 1 开始
31.   {
32.       int k = arc_table[i];
33.       k = 255 - k;
34.       k %= 255; //m_table 的取值范围为 [0, 254]
35.       inverse_table[i] = table[k];
36.   }
37.
38.   printf("(03)对应: %d\n\n", arc_table[3]);
39.   printf("(253)对应: %d\n\n", arc_table[253]);
40.   printf("结果多项式对
应: %d\n\n", table[(arc_table[3] + arc_table[253]) % 255
]);
41.   printf("(03)逆元对应: %d\n\n", inverse_table[3]);
42.   system("pause");
43.   return 0;
44.}

```

说明：由实验原理，正表即指数表，生成元“02”对应生成多项式为 α ，执行循环体计算 $\alpha^i \bmod(f(\alpha))$ ，其中不可约多项式 $f(\alpha) = \alpha^8 + \alpha^6 + \alpha^5 + \alpha + 1$ 。利用递归思路，依次按 $GF(2^8)$ 域上的乘法计算正表 table。再根据正表建立对数表，即反表 arc_table，即可实现根据查表快速计算域上的多项式乘法。另根据实验原理，只需对下标进行模运算，即可建立逆元表，实现查表快速计算域上的多项式的逆。

(2) 测试用例

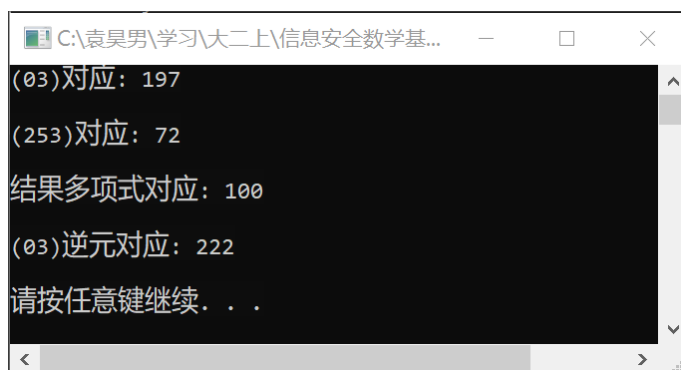
注：求 $f(x) \cdot g(x)$ 以及 $h(x)$ 的逆元。

序号	输入 $f(x)$ 对应十进制码 $g(x)$ 对应十进制码 $h(x)$ 对应十进制码	预期输出 $f(x) \cdot g(x)$ 、 $h^{-1}(x)$ 对应十进制码	实际输出 $f(x) \cdot g(x)$ 、 $h^{-1}(x)$ 对应十进制码
1	$[f(x)] = 03$ $[g(x)] = 253$ $[h(x)] = 03$	$[f(x) \cdot g(x)] = 100$ $[h^{-1}(x)] = 222$	$[f(x) \cdot g(x)] = 100$ $[h^{-1}(x)] = 222$
2	$[f(x)] = 14$ $[g(x)] = 233$ $[h(x)] = 10$	$[f(x) \cdot g(x)] = 178$ $[h^{-1}(x)] = 37$	$[f(x) \cdot g(x)] = 178$ $[h^{-1}(x)] = 37$
3	$[f(x)] = 67$ $[g(x)] = 111$	$[f(x) \cdot g(x)] = 112$ $[h^{-1}(x)] = 160$	$[f(x) \cdot g(x)] = 112$ $[h^{-1}(x)] = 160$

	$[h(x)] = 56$		
--	---------------	--	--

(3) 运行截图

a) 测试用例 1

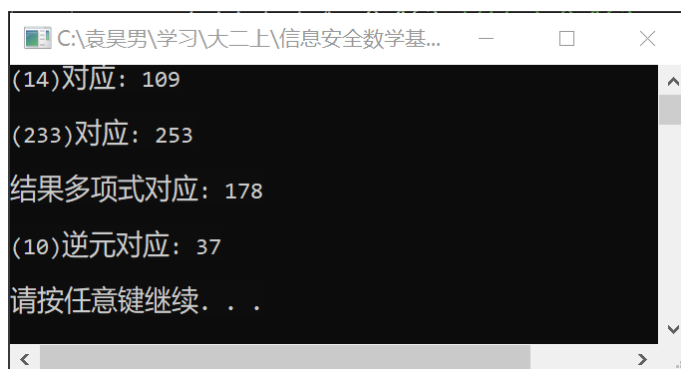


```

C:\袁昊男\学习\大二上\信息安全数学基...
(03)对应: 197
(253)对应: 72
结果多项式对应: 100
(03)逆元对应: 222
请按任意键继续. . .

```

b) 测试用例 2

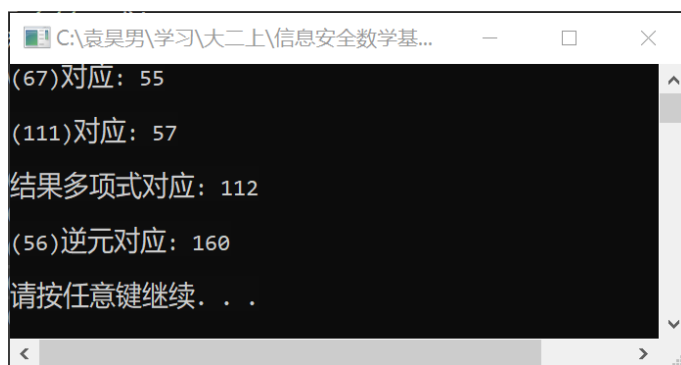


```

C:\袁昊男\学习\大二上\信息安全数学基...
(14)对应: 109
(233)对应: 253
结果多项式对应: 178
(10)逆元对应: 37
请按任意键继续. . .

```

c) 测试用例 3



```

C:\袁昊男\学习\大二上\信息安全数学基...
(67)对应: 55
(111)对应: 57
结果多项式对应: 112
(56)逆元对应: 160
请按任意键继续. . .

```

九、 总结及心得体会：

本实验是有关 $GF(2^8)$ 域上的多项式运算的实现，实验的内容较多，且此部分的理论学时较少，对初学者来说理解较困难，再加上对 C 语言中的位运算不是特别熟悉，在编程实现时有不小的困难。首先通过多项式基的表示，建立了 8 位二进制码与多项式之间的一一对应关系，使得多项式乘法、求逆转化为对多项式系数的操作，简化了后续的计算过程；为了更快速的计算多项式乘法以及逆元，引入了指数对数表，即事先由本原元多项式生成整个多项式域，每一个幂次对应一个计算结果，

使得在以后计算多项式乘法时，只需查指数对数表，找到对应的幂次结果，直接对指数进行模加法运算即可。还可以此原理建立逆元表，方便进行多项式逆元的计算。

在编程中，通过与实验三中“扩展的欧几里得算法”的对比，我发现 Python 较其他语言在密码学的相关算法的实现上有先天的优势，代码量远远少于 C 语言，且简明易懂，在以后的学习、编程中，尽量采用 Python 来提高工作效率。

十、 对本实验过程及方法、手段的改进建议：

本实验中多项式的相关内容可以在理论教学时适当增加课时数，且本课程的实验统一安排在期末进行学生完成的时间压力较大。

报告评分：

指导教师签字：