**电子科技大学信息与软件工程学院**

**标 准 实 验 报 告**

**（实验）课程名称 信息安全数学基础实验**

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名： 陆圣珩 学 号：2017221302009 指导教师： 陈大江**

**实验地点：3教410 实验时间：12.15**

**一、实验室名称： 网络安全实验室**

**二、实验项目名称：有限域上运算的实现**

**三、实验学时：2学时**

**四、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（28）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）。

**五、实验内容：**

实现28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，从而实现28域上元素乘法运算和逆元运算。更进一步，构造指数对数表，从而通过查表实现28域上元素乘法运算和逆元运算。

**六、实验步骤：**

1. 熟悉28域中元素的表示；
2. 熟悉多项式模乘运算和求逆运算；
3. 进行算法需求分析；
4. 制定算法流程图；
5. 学习相关的类及相关成员的建立；
6. 根据算法需求分析和算法流程图编写实现代码；
7. 编译并运行。

**七、实验代码**

|  |
| --- |
| **""" 名字：Mod(a,b,n),a,b表示列表，长度都为n 功能：返回a模b的列表 """****def** Mod(a, b, n):  result = a[:]  d = b[:]  a\_len = n  b\_len = n  **for** i **in** range(n):  **if** a[i] != 0:  **break** a\_len -= 1  **for** i **in** range(n):  **if** b[i] != 0:  **break** b\_len -= 1  **if** a\_len >= b\_len:  **for** i **in** range(b\_len, a\_len):  d.append(0)  **for** i **in** range(a\_len-1, b\_len-2, -1):  **if** result[n-i-1] == 1:  result=addsub(result, d[i - b\_len + 1:n + i - b\_len + 1], n)  **return** result   **""" 名字：AddSub(a,b,n),a,b为列表，长度为n 功能：返回a减b或者a加b 说明：无论是加运算,还是减运算，在模2运算中算法相同 """   def** addsub(a, b, n):  c=[]  **for** i **in** range(n):  c.append(a[i] ^ b[i])  **return** c   **""" 名字：Multiply(a,b,n),a,b为列表，长度为n 功能：返回a乘b的元素 """   def** multiply(a, b, n, mod):  c = a[:]  **while** n < len(mod):  c.insert(0, 0)  b.insert(0, 0)  n += 1  result = [0 **for** i **in** range(n)]  **for** i **in** range(n-1, -1, -1):  **if** b[i] == 1:  result = addsub(result, c, n)  c.append(0)   c.pop(0)  c = list(Mod(c, mod, n))  **return** result   **""" 名字：Divide(a,b,n),a,b为列表，长度为n 功能：返回a整除b后的多项式 """   def** divide(a, b, n):  c = a[:]  d = b[:]  result = [0 **for** i **in** range(n)]  a\_len = n  b\_len = n  **for** i **in** range(n):  **if** a[i] != 0:  **break** a\_len -= 1  **for** i **in** range(n):  **if** b[i] != 0:  **break** b\_len -= 1  **if** a\_len >= b\_len:  **for** i **in** range(b\_len, a\_len):  d.append(0)  **for** i **in** range(a\_len-1, b\_len-2, -1):  **if** c[n-i-1] == 1:  result[n-i+b\_len-2] = 1  c=addsub(c, d[i - b\_len + 1:n + i - b\_len + 1], n)  **return** result     **""" 名字：getRes(a,b,n,mod),a,b为列表，长度为n 功能：求多项式a关于b的逆元  原扩展欧几里得算法 def OGLD\_pro(a,b):  if (b==0):  return {'d':a,'x':1,'y':0}  else:  t=OGLD\_pro(b,a%b)  temp=t['x']  t['x']=t['y']  t['y']=temp-int(a/b)\*t['y']  return t """   def** getres(a, b, n, mod):  **if** max(b) == 0:  x = [0 **for** i **in** range(n-1)]  x.append(1)  **return** {**'d'**: a[:], **'x'**: x, **'y'**: [0 **for** i **in** range(n)]}  **else**:  t = getres(b, Mod(a, b, n), n, mod)  temp = list(t[**'x'**])  t[**'x'**] = list(t[**'y'**])  t[**'y'**] = addsub(temp, multiply(divide(a, b, n), t[**'y'**], n, mod), n)    **return** t   **""" 名字：ExpOfMod(a,k,myMod,n):列表a,k为指数，myMod为要模的多项式，n为列表长度 功能：模的指数运算，重复平方算法  """   def** ExpOfMod(a,k,mod,n):  result=[0 **for** i **in** range(n-1)]  result.append(1)  **if** k != 0:  A = a[:]  **if** k % 2 == 1:  result = a[:]  k = k // 2  **while** k != 0:  A = multiply(A, A, n, mod)  **if** k % 2 == 1:  result = multiply(A, result, n, mod)  k = k // 2  **return** result   **""" 名字：transList(lst),lst为列表 功能:用于把列表中二进制数转化为10进制 """   def** translist(lst):  result=0  **for** i **in** lst:  result \*= 2  result += i    **return** result   **""" 名字：基准，即要模得多项式，用多项式基表示 功能：作为全局变量 注意：多项式一定不能拆分，即不可约多项式 """** myMod = [1,0,1,1,0,0,0,1,1] testMod =[1,0,0,1,1]   a = [0, 0, 0, 0, 0, 0, 1, 1, 1] b = [0, 1, 0, 1, 0, 1, 1, 0, 1] print(**"a="**, a) print(**"b="**, b) print(**"a+b=a-b="**, addsub(a, b, len(a))) print(**"b/a="**, divide(b, a, len(a))) print(**"a\*b="**, multiply(a, b, len(a), myMod)) t = getres(a, myMod, 9, myMod) a\_ = t[**'x'**] print(**"a得逆元为："**, a\_)   base = [0, 0, 0, 0, 0, 0, 0, 1, 0] ExpList = [0 **for** i **in** range(256)] LogList = [0 **for** i **in** range(256)] LogList[0] = 255 **for** i **in** range(255):  index = translist(ExpOfMod(base, i, myMod, len(base)))  ExpList[i] = index  LogList[index] = i print(**"\n\t\t\t\t\t指数对数表\n\n"**) i = 0 **while** i < 256:  **while True**:  **if** i % 16 == 0:  print(**"{:8s}"**.format(**"序号"**), end=**' '**)  print(**"{:4d}"**.format(i), end=**' '**)  **if** (i+1)%16 == 0:  print(**'\n'**)  i = i-15  **break** i += 1  **while True**:  **if** i % 16 == 0:  print(**"{:9s}"**.format(**"(02)^i"**), end=**' '**)  print(**"{:4d}"**.format(ExpList[i]), end=**' '**)  **if** (i+1) % 16 == 0:  print(**'\n'**)  i = i-15  **break** i += 1  **while True**:  **if** i % 16 == 0:  print(**"{:7s}"**.format(**"log(02)^i"**), end=**' '**)  print(**"{:4d}"**.format(LogList[i]), end=**' '**)  **if** (i+1) % 16 == 0:  print(**'\n\n'**)  i += 1  **break** i += 1 |

**八、实验数据及结果分析：**

|  |
| --- |
|  |

**九、实验结论：**

通过python实现编程要求，测试结果正确。

**十、总结及心得体会：**

整个实验的难度非常大，在几个同学的共同努力下才勉强完成本次实验。有限域的运算本身需要一定的背景知识了解，必须在数学基础完备的情况下才能完成本次编程，而整个程序的代码也很长，需要细致的编程和测试。

**十一、对本实验过程及方法、手段的改进建议：**

整体编程难度过大，部分要求不明确，可以在实验要求中要求的更加细致准确一些。

**报告评分：**

**指导教师签字：**