# 汇编期末实验说明文档

# 任务一

小组成员：

杨斌 12335241 曾馨缘 12335267 钟腾 12335287

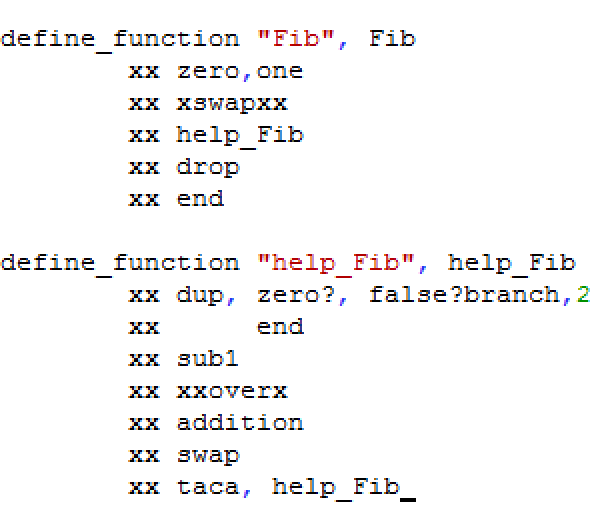
刘人翡 12335142 王彪 12335193

本次实验中，我们对助教提供的5th-instar程序进行了修改。首先我们发现助教在此段程序中用了两种宏来定义函数。一种是定义primitive函数的，一种是定义普通函数。其中包括加减乘除求余，大小比较函数，对栈元素的操作over,tuck,swap,dup,drop，以及输入输出函数；save，fetch函数等。

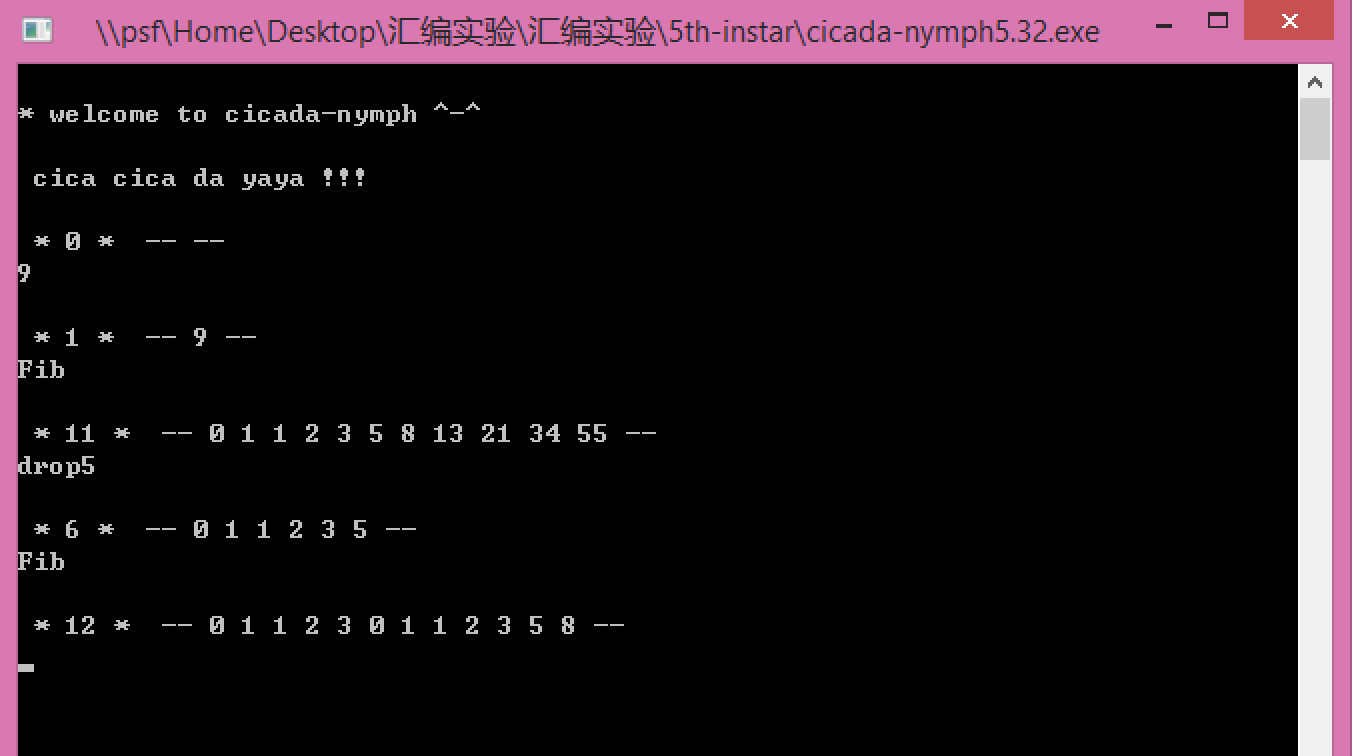
在5th-instar这段程序中我们一共实现了以下4个功能。

1. **求斐波那契数列**

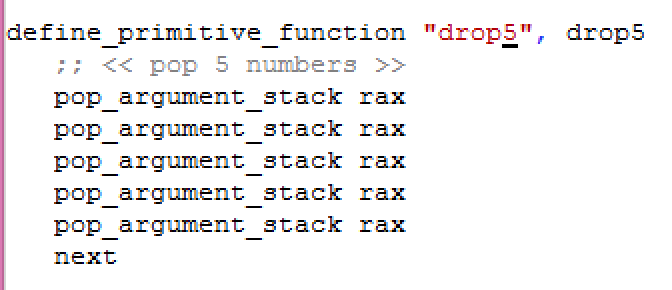
首先为了求出斐波那契数列，我们定义了函数“Fib”以及函数“help\_Fib”，具体代码如下：



所得出的运算结果如下图所示：



其中drop5的函数代码如下：

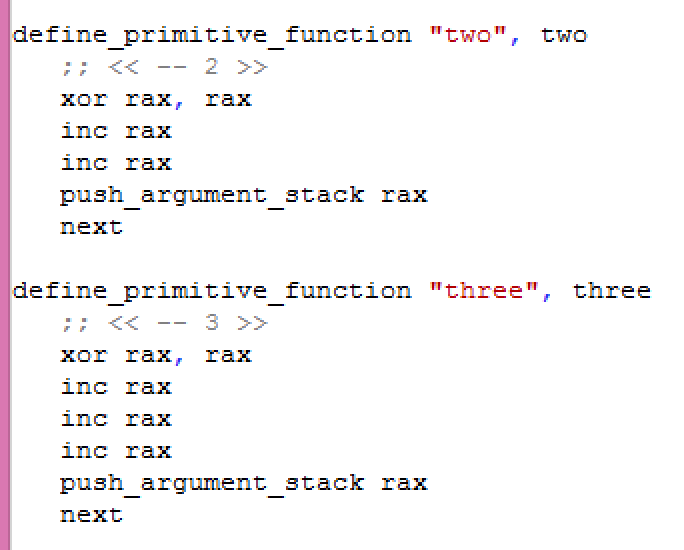


在第一次运行Fib函数时，我们得到1+9+1=11个栈数的斐波那契数列，接下来借鉴老师所写的“drop”函数，写了一个“drop5”函数，将斐波那契数列最后5个数弹出。再次运行Fib函数得到0 1 1 2 3后的“0 1 1 2 3 5 8”7个栈数的斐波那契数列。结果验证了我们的“Fib”函数能够实现求斐波那契数列的功能。

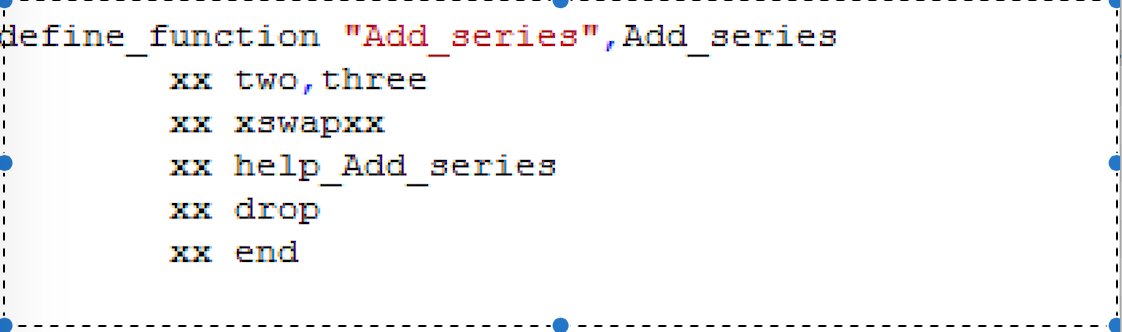
1. **广义连加**

受求斐波拉切数列思想启发，我们将其推广到任意两个数为初值的广义连加。首先定义了“two”和“three”的两个primitive\_function，并编写了“Add\_series”函数以实现我们的想法。同时，为了避免函数过于臃杂，我们还需要定义一个辅助函数help\_Add\_series帮助实现运算。以下为函数代码：

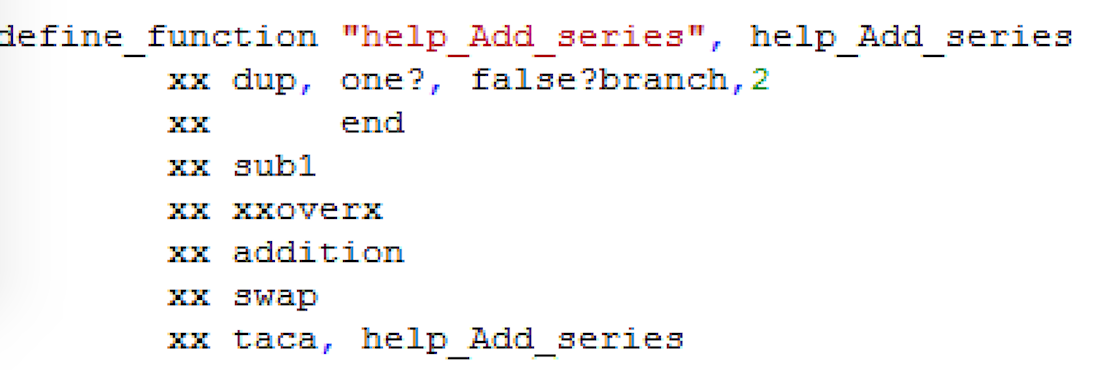
two,three 的primitive\_function：



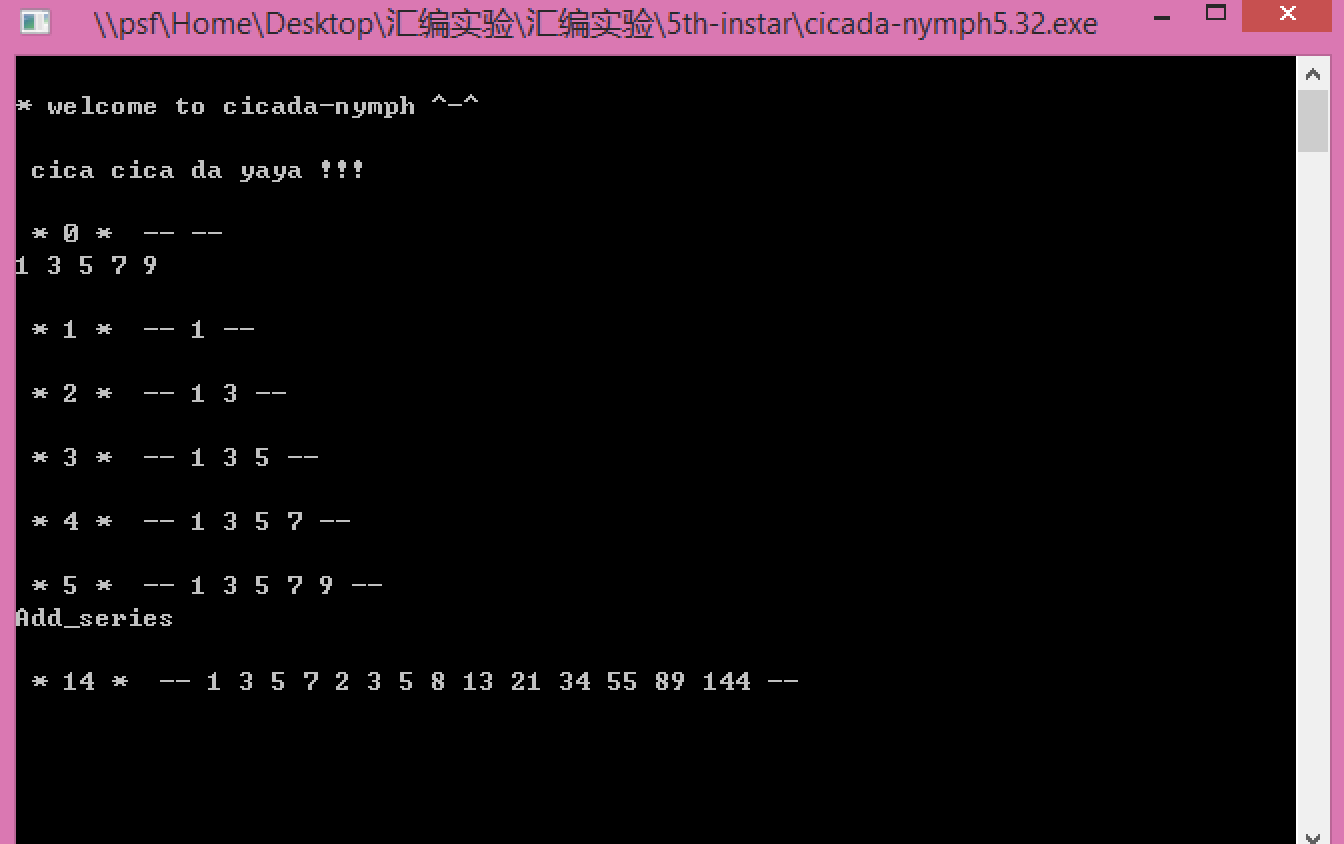
Add\_series:



help\_Add\_series:



以下为函数运行结果：

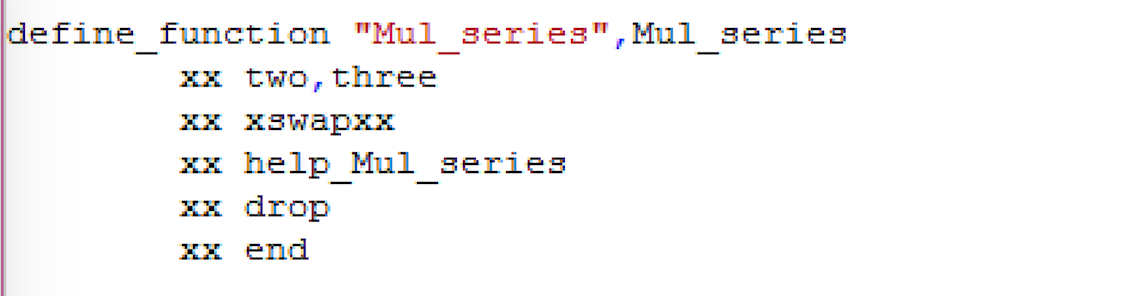


首先我们输入数列：1 3 5 7 9将这5个数分配到5个栈位中。接着运行Add\_series得到5+9=14个栈位的数列。因为Two和Three两个primitive\_function函数的目的是在栈尾加上2和3。因此我们运行得到的结果是将数列1 3 5 7 9的最后一位数9 弹出之后以2和3 开头的连加数列。即输出 1 3 5 7 2 3 5 8 13 21 34 55 89 144。

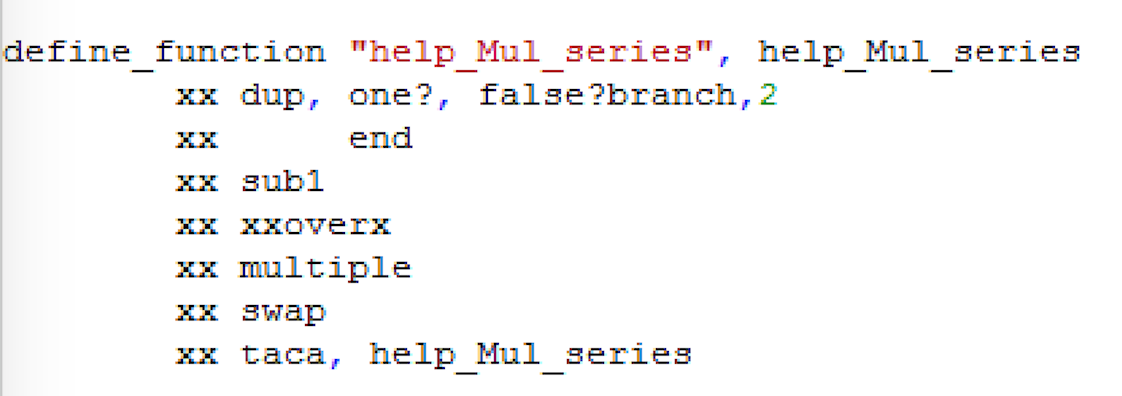
1. **广义连乘**

同样的道理，我们将其连加的思想推广到任意两个数的广义连乘。首先同样利用“two”和“three”的两个primitive\_function，并编写了“Mul\_series”函数以实现连乘功能。同时，为了避免函数过于臃杂，我们同样定义一个辅助函数help\_Mul\_series帮助实现运算。以下为函数代码：

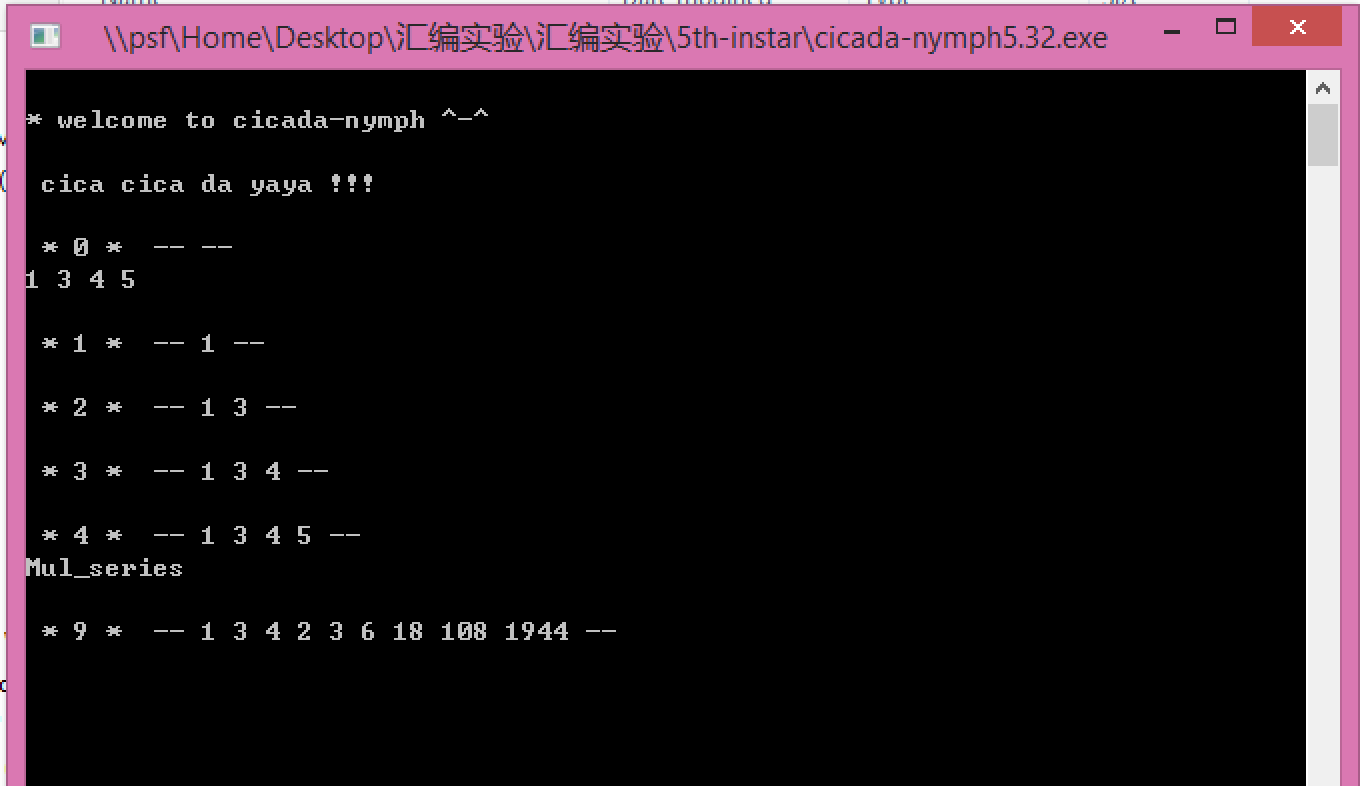
Mul\_series:



help\_Mul\_series:

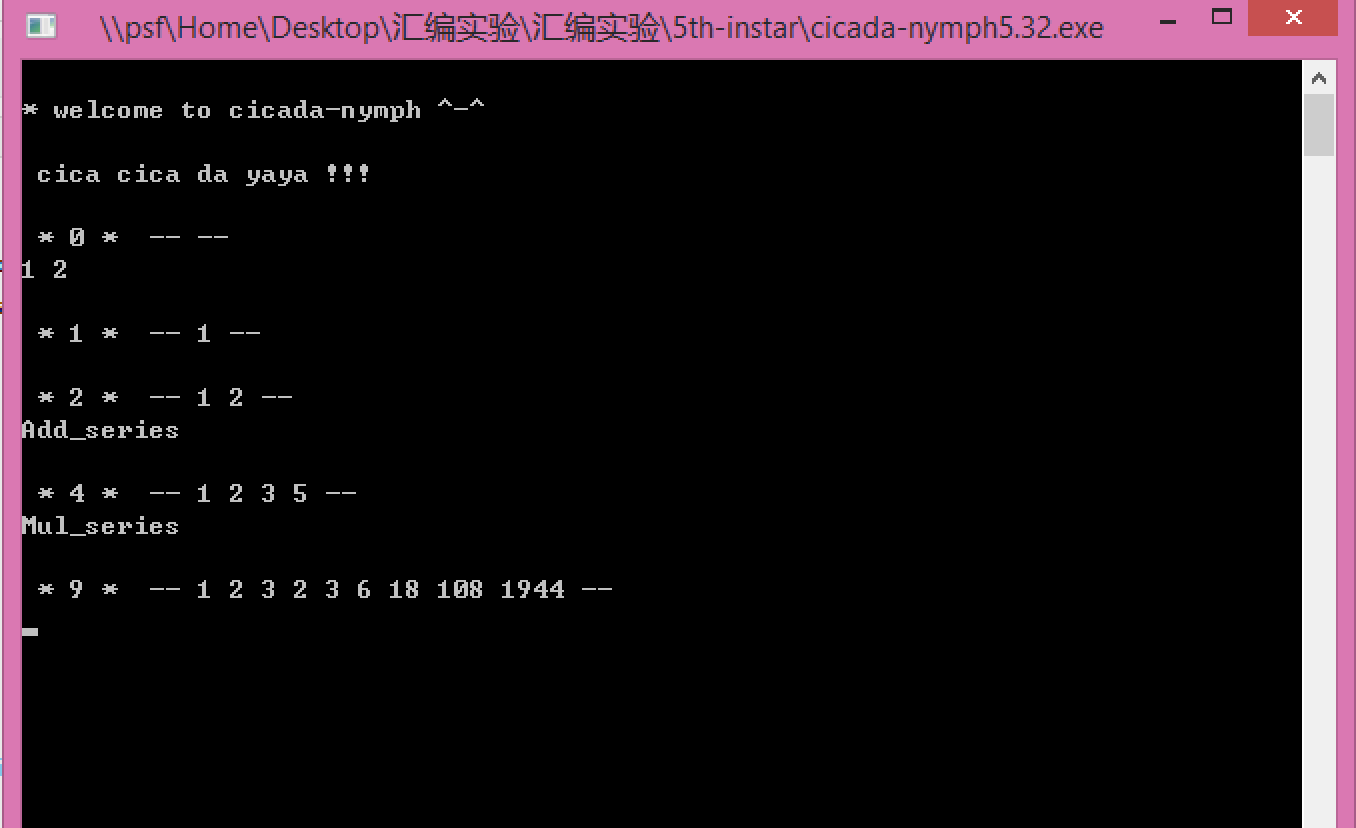


以下为函数运行结果：



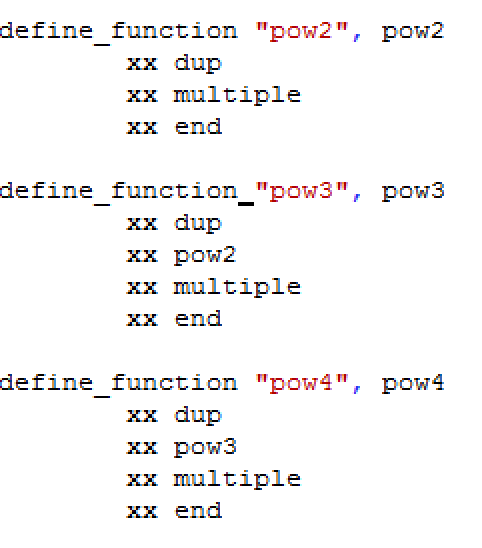
首先我们输入数列：1 3 4 5将这4个数分配到4个栈位中。接着运行Mul\_series得到4+5=9个栈位的数列。因为Two和Three两个primitive\_function函数的目的是在栈尾加上2和3。因此我们运行得到的结果是将数列1 3 4 5 的最后一位数5 弹出之后以2和3 开头的连乘数列。即输出 1 3 4 2 3 6 18 108 1944。

我们同时实现连加及连乘功能得到下面运算结果：

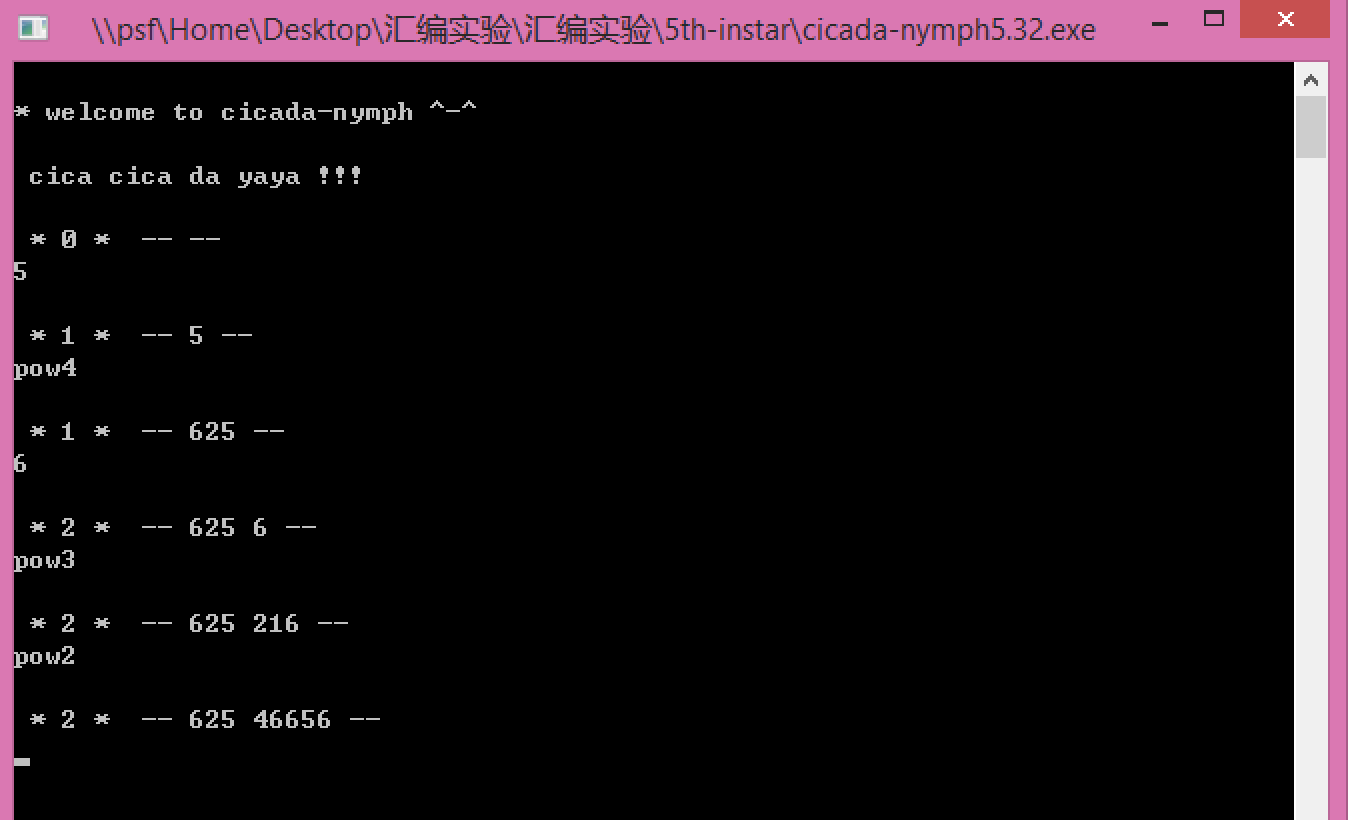


1. **2，3，4次方的幂次运算**

为了实现幂次运算的功能，以2次方为例，我们定义了“pow2”函数，并以此类推得到3，4次方运算函数“pow3”、“pow4”（期中pow4、pow3分别调用了之前的函数），函数代码如下：



以下为程序运算结果：



pow2、pow3、pow4分别返回栈中最后一个数的2、3、4次方。

5^4=625

6^3=216

216^2=46656