很有意思的题目
又学到了许多新姿势
首先看到最后一个输入正确的情况

```
v13 = 0i64;
do
{
  if ( a1234567890Qwer[outputString[v13] % 23] != *(_BYTE *)(v13 + 0x140003478i64) )// (_@4620!08!6_0*0442!@186%%0@3=66!!974*3234=&
    _exit(v12);
  if ( a1234567890Qwer[outputString[v13] / 23] != *(_BYTE *)(v13 + 0x140003438i64) )// 55565653255552225565565555243466334653663544
    _exit(v12 * v12);
  ++v12;
  ++v13;
}
while ( v12 < 0x3E );
printf("flag{MD5(your input)}\n");
```

可以写个脚本跑出outputstring的值

```
check1 = list(b'(_@4620!08!6_0*0442!@186%%0@3=66!!974*3234=&0^3&1@=&0908!6_0*&')
check2 = list(b'555656532555522255655655552434663346536635444265655555255555222')
index = list(b"1234567890-=!@#$%^&*()_+qwertyuiop[]QWERTYUIOP{}asdfghjkl;'ASDFGHJKL:\"ZXCVBNM<>?
outputstring = []
for i in range(62):
    a = check1[i]
    b = check2[i]
    for j in range(10,127):
        if index[j%23]==a and index[j//23]==b:
            outputstring.append(j)
            break
print(bytes(outputstring))
print(len(outputstring))
```

```
PS C:\Users\wsxk> & e:/python/virtual_environment/z3_environment/Scripts/pytho
E/process_data.py
b'private: char * __thiscall R0Pxx::My_Aut0_PWN(unsigned char *)'
62
```

可以看出这是一个很明显的c++声明标识符

```
}
UnDecorateSymbolName(v5, outputString, 0x100u, 0);// outputstring约等于input
v9 = -1i64;
do
  ++v9;
```

上述的声明标识符是通过这个函数将输入值解修饰来的
https://docs.microsoft.com/en-us/windows/win32/api/dbghelp/nf-dbghelp-undecoratesymbolname
上述地址可以看道UnDecorateSymbolName的作用
接下来我们需要恢复其修饰（即c++编译器在将编译时对函数的修饰）

原值：即原函数名：My_Aut0_PWN
参考资料1

可以知道第二个参数为未修饰的名字，第三个参数为长度，第四个参数为0表示完全修饰，第一个参数为输出地址

参考资料2

c++函数名的修饰更为复杂，提供的信息也更为丰富。
无论 __cdecl，__fastcall还是__stdcall调用方式，函数修饰都是以一个“?”开始，后面紧跟函数的名字。再后面是

?My_Aut0_PWN

参考资料3

对于C++的类成员函数(其调用方式是thiscall)，函数的名字修饰与非成员的C++函数稍有不同，首先就是在函数名字和

?My_Aut0_PWN@R0Pxx

参考资料4

其次是参数表的开始标识不同，公有（public）成员函数的标识是“@@QAE”，保护（protected）成员函数的标识是 “@

?My_Aut0_PWN@R0Pxx@@AAE

参考资料5

添加参数信息
参数表的拼写代号如下：
X-void
D-char
E-unsigned char
F-short
H-int
I-unsigned int
J-long
K-unsigned long (DWORD)
M-float
N-double
_N-bool
U-struct
…
指针的方式有些特别。用PA表示指针，用PB表示const类型的指针。

添加参数时首先添加返回值，然后添加输入参数
返回值为char * 所以为PAD
?My_Aut0_PWN@R0Pxx@@AAEPAD
参数是unsigned char * 即 PAE
?My_Aut0_PWN@R0Pxx@@AAEPAD即为最终结果

然后再往前，有一个二叉树形式的置换表

十六进制												ASCII
5A	30	40	74	52	41	45	79	75	50	40	78	Z0@tRAEyuP@x
41	41	41	3F	4D	5F	41	30	5F	57	4E	50	AAA?M_A0_WNP
78	40	40	45	50	44	50	00					x@@EPDP.

因为是置换，只需把?My_Aut0_PWN@R0Pxx@@AAEPAD输入程序调试即可得到其置换后的结果
置换后的结果即为正确输入（再置换能得到?My_Aut0_PWN@R0Pxx@@AAEPAD）

所以输入是Z0@tRAEyuP@xAAA?M_A0_WNPx@@EPDP