# homework1

```
push    ebp
mov     ebp, esp
push    offset Buffer   ; "Input the password:"
call    ds:puts
add     esp, 4
push    offset String   ; char
push    offset Format   ; "%10s"
call    sub_401040
add     esp, 8
push    offset String   ; String
call    ds:atoi
add     esp, 4
imul    eax, 5
cmp     eax, 181Ah
```

由IDA反汇编后得到的主函数如上，后面剩余的函数部分是根据比较结果给出正确和错误提示并推出进程。

实际上，找答案主要看以下几行：

```
call    ds:atoi
add     esp, 4
imul    eax, 5
cmp     eax, 181Ah
```

调用`ds:atoi`函数，将用户输入字符串转为数字，由于`eax`是返回值默认存储器，所以用户输入得到的数字被存在了`eax`中。调用了一个函数，所以`ebp`加4，随后`eax`自乘5，与16进制数181A进行比较，如果等于181A则密码正确。181A转为十进制应该为`6170`,则正确的输入应该为`6170/5`，即`1234`

# homework2

## 结果与C代码

输出结果为：`fffce42f`

c代码：

```
#include<stdio.h>
int main(){
    int x=3,y=4,z=0;
    while(1)
    {   if(z>=10)
    {
        break;
    }
        x=x+y;
        y=y+x;
        z++;
    }
    if(x>=y)
```

```
    {if(y-x>3)
    {y=x*y;
    x=x+y;
    }
    }
    else{
        x=x*y;
        y=y-x;
    }
    printf("%x\n",x^y);
    return 0;
}
```

# 分析思路：

```
.text:00401080 var_C           = dword ptr -0Ch
.text:00401080 var_8           = dword ptr -8
.text:00401080 var_4           = dword ptr -4


.text:00401080                 push    ebp
.text:00401081                 mov     ebp, esp
.text:00401083                 sub     esp, 0Ch
.text:00401086                 mov     [ebp+var_4], 3
.text:0040108D                 mov     [ebp+var_8], 4
.text:00401094                 mov     [ebp+var_C], 0
.text:0040109B                 jmp     short loc_4010A6
```

这段代码表示存了三个变量，值分别为3，4，0。随后进入下一个部分。

将之记为x,y,z。

```
.text:0040109D loc_40109D:
.text:0040109D                 mov     eax, [ebp+var_C]
.text:004010A0                 add     eax, 1
.text:004010A3                 mov     [ebp+var_C], eax
.text:004010A6 loc_4010A6:
.text:004010A6                 cmp     [ebp+var_C], 0Ah
.text:004010AA                 jge     short loc_4010C0
.text:004010AC                 mov     ecx, [ebp+var_4]
.text:004010AF                 add     ecx, [ebp+var_8]
.text:004010B2                 mov     [ebp+var_4], ecx
.text:004010B5                 mov     edx, [ebp+var_8]
.text:004010B8                 add     edx, [ebp+var_4]
.text:004010BB                 mov     [ebp+var_8], edx
.text:004010BE                 jmp     short loc_40109D
```

这是一个循环，每次先判断z的值是否大于等于10，大于10则跳出。

循环中

的内容为`x=x+y,y=x+y,z++`三句

```
.text:004010C0 loc_4010C0:
.text:004010C0                 mov     eax, [ebp+var_4]
.text:004010C3                 cmp     eax, [ebp+var_8]
```

```
.text:004010C6                 jge     short loc_4010DD
.text:004010C8                 mov     ecx, [ebp+var_4]
.text:004010CB                 imul    ecx, [ebp+var_8]
.text:004010CF                 mov     [ebp+var_4], ecx
.text:004010D2                 mov     edx, [ebp+var_8]
.text:004010D5                 sub     edx, [ebp+var_4]
.text:004010D8                 mov     [ebp+var_8], edx
.text:004010DB                 jmp     short loc_4010FB
.text:004010DD loc_4010DD:
.text:004010DD                 mov     eax, [ebp+var_8]
.text:004010E0                 sub     eax, [ebp+var_4]
.text:004010E3                 cmp     eax, 3
.text:004010E6                 jle     short loc_4010FB
.text:004010E8                 mov     ecx, [ebp+var_8]
.text:004010EB                 imul    ecx, [ebp+var_4]
.text:004010EF                 mov     [ebp+var_8], ecx
.text:004010F2                 mov     edx, [ebp+var_4]
.text:004010F5                 add     edx, [ebp+var_8]
.text:004010F8                 mov     [ebp+var_4], edx
```

这一段先对x和y的大小进行比较：

如果x>=y,则进入分支中，再比较y-x和3的关系，如果大于就`y=y*x,x=x+y`。

如果x<y，则`x=x*y,y=y-x`。

```
.text:004010FB                 mov     eax, [ebp+var_4]
.text:004010FE                 xor     eax, [ebp+var_8]
.text:00401101                 push    eax             ; char
.text:00401102                 push    offset Format   ; "%x\n"
.text:00401107                 call    printf
.text:0040110C                 add     esp, 8
.text:0040110F                 mov     eax, 2766h
.text:00401114                 mov     esp, ebp
.text:00401116                 pop     ebp
.text:00401117                 retn
```

最后按照十六进制输出x^y,结束程序。