1、这个题目针对图 *7-5* 中的 *m.o* 和 *swap.o* 模块。对于每个在 *swap.o* 中定义或引用的符号，请指出它是否在模块 *swap.o* 中的 *.symtab* 节中有一个符号表条目。如果是，请指出定义该符号的模块（*swap.o* 或者 *m.o*）、符号类型（局部、全局或者外部）以及它在模块中被分配到的节（*.text*、*.data*、*.bss* 或 *COMMON*）

符号	.symtab条目?	符号类型	在哪个模块中定义	节
buf				
bufp0				
bufp1				
swap				
temp				

code/link/m.c

```
1   void swap();
2
3   int buf[2] = {1, 2};
4
5   int main()
6   {
7       swap();
8       return 0;
9   }
```

code/link/m.c

a) m.c

code/link/swap.c

```
1   extern int buf[];
2
3   int *bufp0 = &buf[0];
4   int *bufp1;
5
6   void swap()
7   {
8       int temp;
9
10      bufp1 = &buf[1];
11      temp = *bufp0;
12      *bufp0 = *bufp1;
13      *bufp1 = temp;
14  }
```

code/link/swap.c

b) swap.c

预览

2、在此题中，REF（x.i）-> DEF（x.k）表示链接器将把模块 *i* 中对符号 *x* 的任意引用与模块 *k* 中 *x* 的定义关联起来。对于下面的每个示例，用这种表示法来说明链接器将如何解析每个模块中对多重定义符号的引用。如果有一个链接时错误（规则1），写“错误”。如果链接器从定义中任意选择一个（规则3），则写“未知”。

A.

```
/* Module 1 */
int main()
{
}
```

```
/* Module 2 */
int main;
int p2()
{
}
```

(a) REF(main.1) → DEF(_____._____)

(b) REF(main.2) → DEF(_____._____)

B.

```
/* Module 1 */
void main()
{
}
```

```
/* Module 2 */
int main = 1;
int p2()
{
}
```

(a) REF(main.1) → DEF(_____._____)

(b) REF(main.2) → DEF(_____._____)

C.

```
/* Module 1 */
int x;
void main()
{
}
```

```
/* Module 2 */
double x = 1.0;
int p2()
{
}
```

3、下图为可执行文件 *prog* 已重定位的 *.text* 节。原始的 *C* 代码在教材的图 *7-1* 中

```
1   00000000004004d0 <main>:
2     4004d0:  48 83 ec 08           sub    $0x8,%rsp
3     4004d4:  be 02 00 00 00        mov    $0x2,%esi
4     4004d9:  bf 18 10 60 00        mov    $0x601018,%edi    %edi = &array
5     4004de:  e8 05 00 00 00        callq  4004e8 <sum>      sum()
6     4004e3:  48 83 c4 08           add    $0x8,%rsp
7     4004e7:  c3                    retq

8   00000000004004e8 <sum>:
9     4004e8:  b8 00 00 00 00        mov    $0x0,%eax
10    4004ed:  ba 00 00 00 00        mov    $0x0,%edx
11    4004f2:  eb 09                 jmp    4004fd <sum+0x15>
12    4004f4:  48 63 ca              movslq %edx,%rcx
13    4004f7:  03 04 8f              add    (%rdi,%rcx,4),%eax
14    4004fa:  83 c2 01              add    $0x1,%edx
15    4004fd:  39 f2                 cmp    %esi,%edx
16    4004ff:  7c f3                 jl     4004f4 <sum+0xc>
17    400501:  f3 c3                 repz retq
```

A. 第5行中对sum的重定位引用的十六进制地址是多少?

> 0x4004df

B. 第5行中对sum的重定位引用的十六进制值是多少?

> 0x5

4、考虑目标文件 *m.o* 中对 *swap* 函数的调用（教材图 *7-5* 所示）

```
9:   e8 00 00 00 00          callq  e <main+0xe>      swap()
```

它的重定位条目如下：

```
r.offset = 0xa
r.symbol = swap
r.type   = R_X86_64_PC32
r.addend = -4
```

现在假设链接器将 *m.o* 中的 *.text* 重定位到地址 *0x4004d0*，将 *swap* 重定位到地址 *0x4004e8*。那么 *callq* 指令中对 *swap* 的重定位引用的值是什么?

> ADDR(S) = ADDR(.text) = 0x4004d0
> ADDR(r.symbol) = ADDR(swap) = 0x4004e8

> refaddr = ADDR(S) + r.offset = 0x4004da
> *refptr = (unsigned)(ADDR(r.symbol) + r.addend - refaddr) = 0xa
> 故引用的值为 0xa