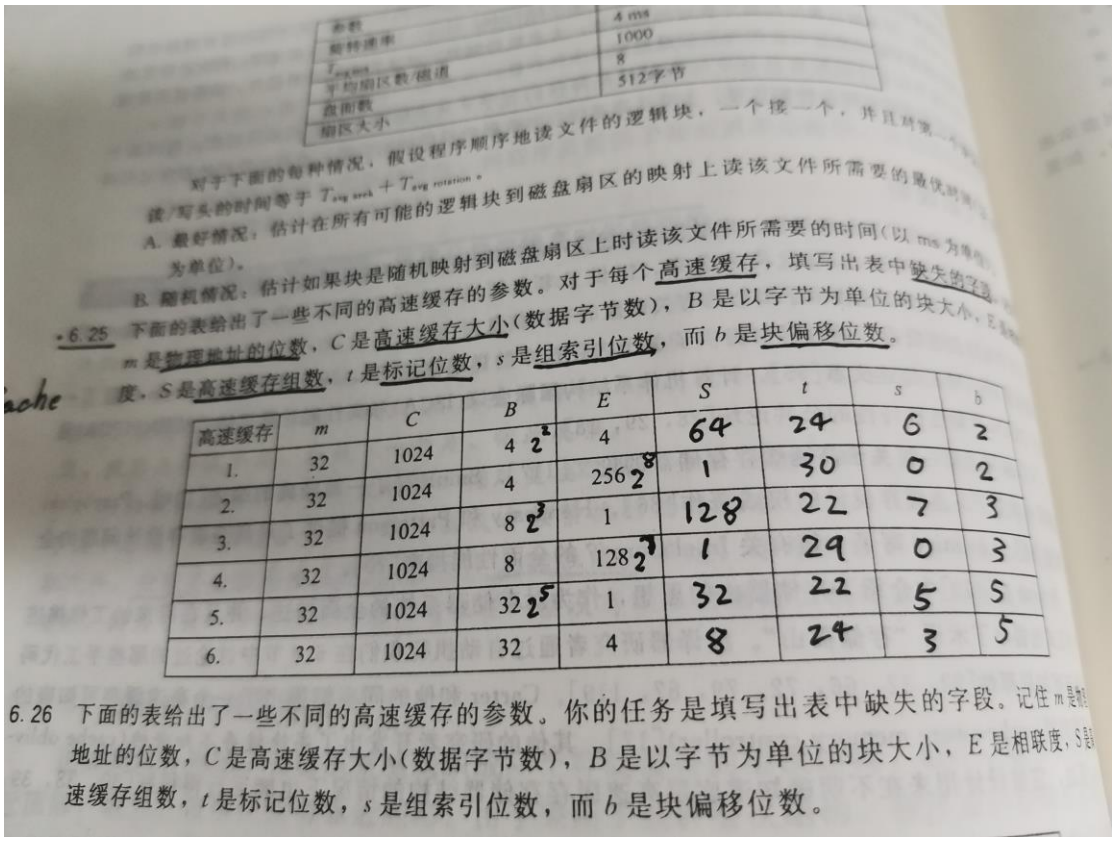
6.25：



参数	
旋转速率	4 ms
$T_{avg\ seek}$	1000
平均扇区数/磁道	8
盘面数	512字节
扇区大小	

对于下面的每种情况，假设程序顺序地读文件的逻辑块，一个接一个，并且对第一个块的读/写头的时间等于 $T_{avg\ seek} + T_{avg\ rotation}$。

A. 最好情况：估计在所有可能的逻辑块到磁盘扇区的映射上读该文件所需要的最优时间（以 ms 为单位）。

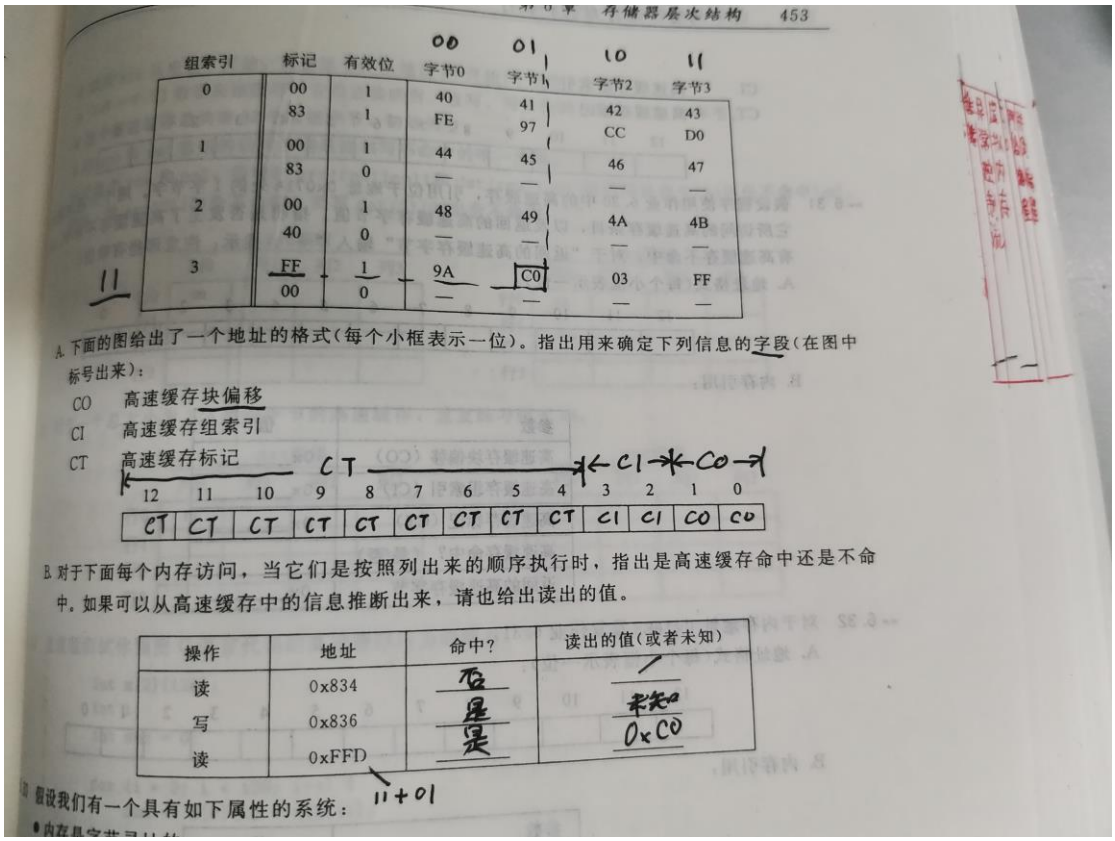
B. 随机情况：估计如果块是随机映射到磁盘扇区上时读该文件所需要的时间（以 ms 为单位）。

·6.25 下面的表给出了一些不同的高速缓存的参数。对于每个高速缓存，填写出表中缺失的字段。m 是物理地址的位数，C 是高速缓存大小（数据字节数），B 是以字节为单位的块大小，E 是相联度，S 是高速缓存组数，t 是标记位数，s 是组索引位数，而 b 是块偏移位数。

高速缓存	m	C	B	E	S	t	s	b
1.	32	1024	4 2^2	4	64	24	6	2
2.	32	1024	4	256 2^8	1	30	0	2
3.	32	1024	8 2^3	1	128	22	7	3
4.	32	1024	8	128 2^7	1	29	0	3
5.	32	1024	32 2^5	1	32	22	5	5
6.	32	1024	32	4	8	24	3	5

6.26 下面的表给出了一些不同的高速缓存的参数。你的任务是填写出表中缺失的字段。记住 m 是物理地址的位数，C 是高速缓存大小（数据字节数），B 是以字节为单位的块大小，E 是相联度，S 是高速缓存组数，t 是标记位数，s 是组索引位数，而 b 是块偏移位数。

6.29：



453

组索引	标记	有效位	00 字节0	01 字节1	10 字节2	11 字节3
0	00	1	40	41	42	43
	83	1	FE	97	CC	D0
1	00	1	44	45	46	47
	83	0	—	—	—	—
2	00	1	48	49	4A	4B
	40	0	—	—	—	—
3	FF	1	9A	C0	03	FF
	00	0	—	—	—	—

A. 下面的图给出了一个地址的格式（每个小框表示一位）。指出用来确定下列信息的字段（在图中标号出来）：

CO 高速缓存块偏移

CI 高速缓存组索引

CT 高速缓存标记

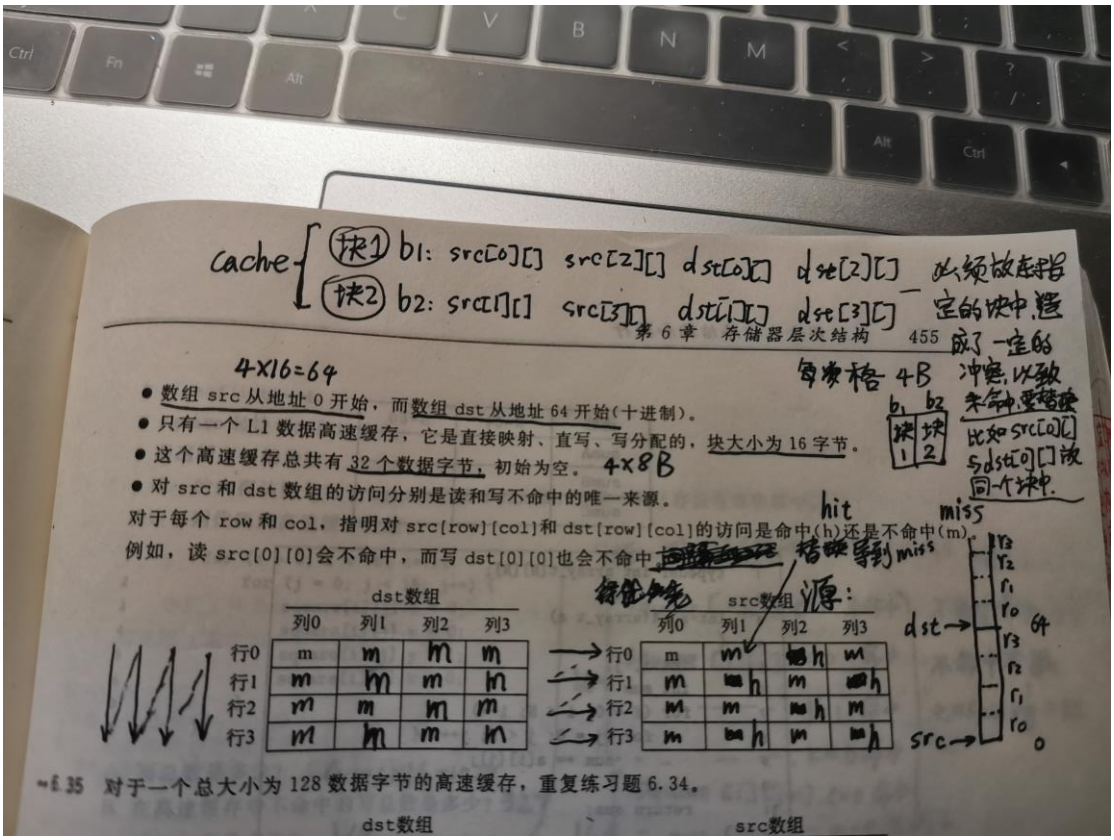
B. 对于下面每个内存访问，当它们是按照列出来的顺序执行时，指出是高速缓存命中还是不命中。如果可以从高速缓存中的信息推断出来，请也给出读出的值。

操作	地址	命中?	读出的值（或者未知）
读	0x834	否	未知
写	0x836	是	0xC0
读	0xFFD	是	

11+01

假设我们有一个具有如下属性的系统：

6.34：



cache { 块1 b1: src[0][] src[2][] dst[0][] dst[2][] 必须放在指定的块中，造成了一定的冲突，以致未命中，要替换
块2 b2: src[1][] src[3][] dst[1][] dst[3][] }

第6章 存储器层次结构 455

4×16=64

每次格 4B

b1 块1 b2 块2

- 数组 src 从地址 0 开始，而数组 dst 从地址 64 开始(十进制)。
- 只有一个 L1 数据高速缓存，它是直接映射、直写、写分配的，块大小为 16 字节。
- 这个高速缓存总共有 32 个数据字节，初始为空。 4×8B
- 对 src 和 dst 数组的访问分别是读和写不命中的唯一来源。

对于每个 row 和 col，指明对 src[row][col]和 dst[row][col]的访问是命中(h)还是不命中(m)。

例如，读 src[0][0]会不命中，而写 dst[0][0]也会不命中

比如 src[0][] 与 dst[0][] 放同一个块中.

hit miss

源：

dst数组	列0	列1	列2	列3
行0	m	m	m	m
行1	m	m	m	m
行2	m	m	m	m
行3	m	m	m	m

src数组	列0	列1	列2	列3
行0	m	m	h	m
行1	m	h	m	h
行2	m	m	h	m
行3	m	h	m	h

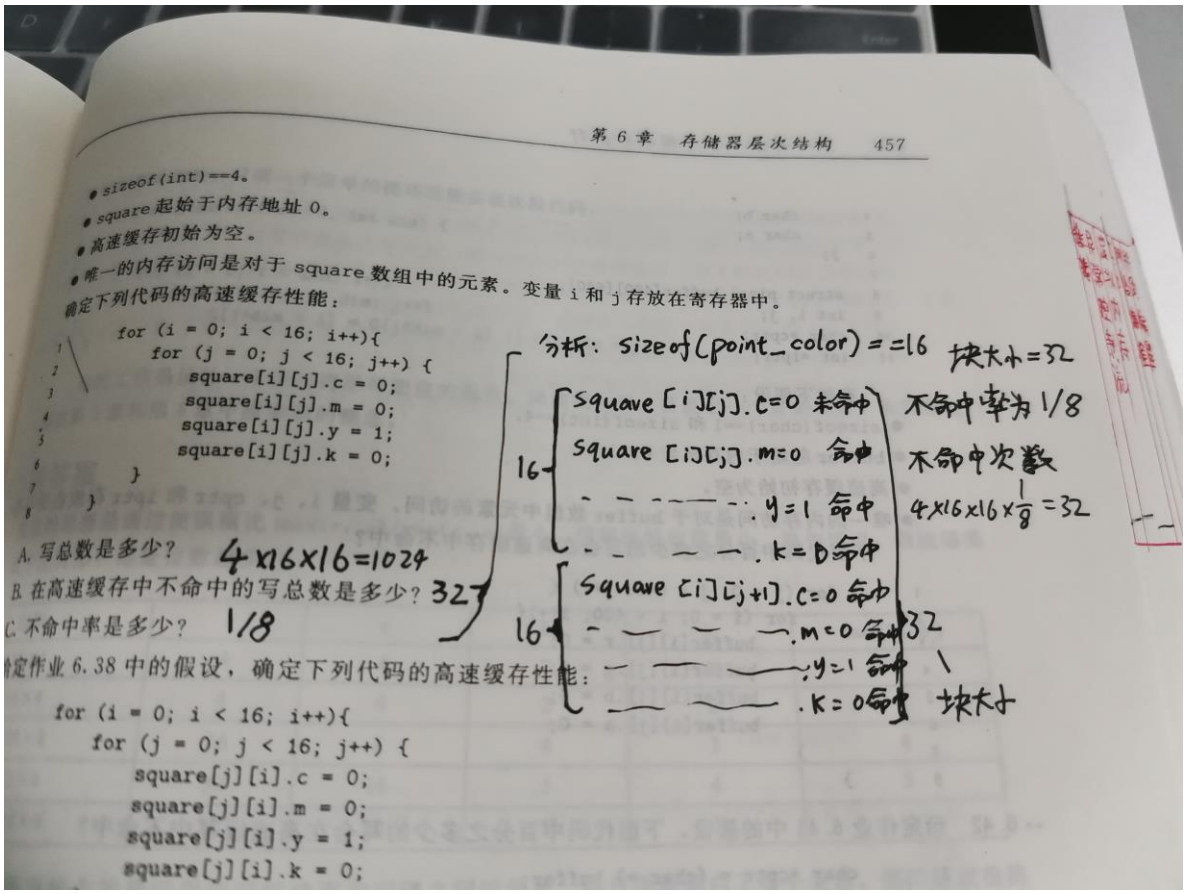
dst → r3 r2 r1 r0 64

src → r3 r2 r1 r0 0

练习题 6.35 对于一个总大小为 128 数据字节的高速缓存，重复练习题 6.34。

dst数组 src数组

6.38：



第6章 存储器层次结构 457

- sizeof(int)==4。
- square 起始于内存地址 0。
- 高速缓存初始为空。
- 唯一的内存访问是对于 square 数组中的元素。变量 i 和 j 存放在寄存器中。

确定下列代码的高速缓存性能：

```
for (i = 0; i < 16; i++){
    for (j = 0; j < 16; j++) {
        square[i][j].c = 0;
        square[i][j].m = 0;
        square[i][j].y = 1;
        square[i][j].k = 0;
    }
}
```

分析：sizeof(point_color) = 16 块大小=32

16 { square[i][j].c=0 未命中 不命中率为 1/8
square[i][j].m=0 命中 不命中次数
.y=1 命中 4×16×16×1/8 = 32
.k=0 命中

16 { square[i][j+1].c=0 命中
.m=0 命中 32
.y=1 命中
.k=0 命中 块大小

A. 写总数是多少？ 4×16×16=1024

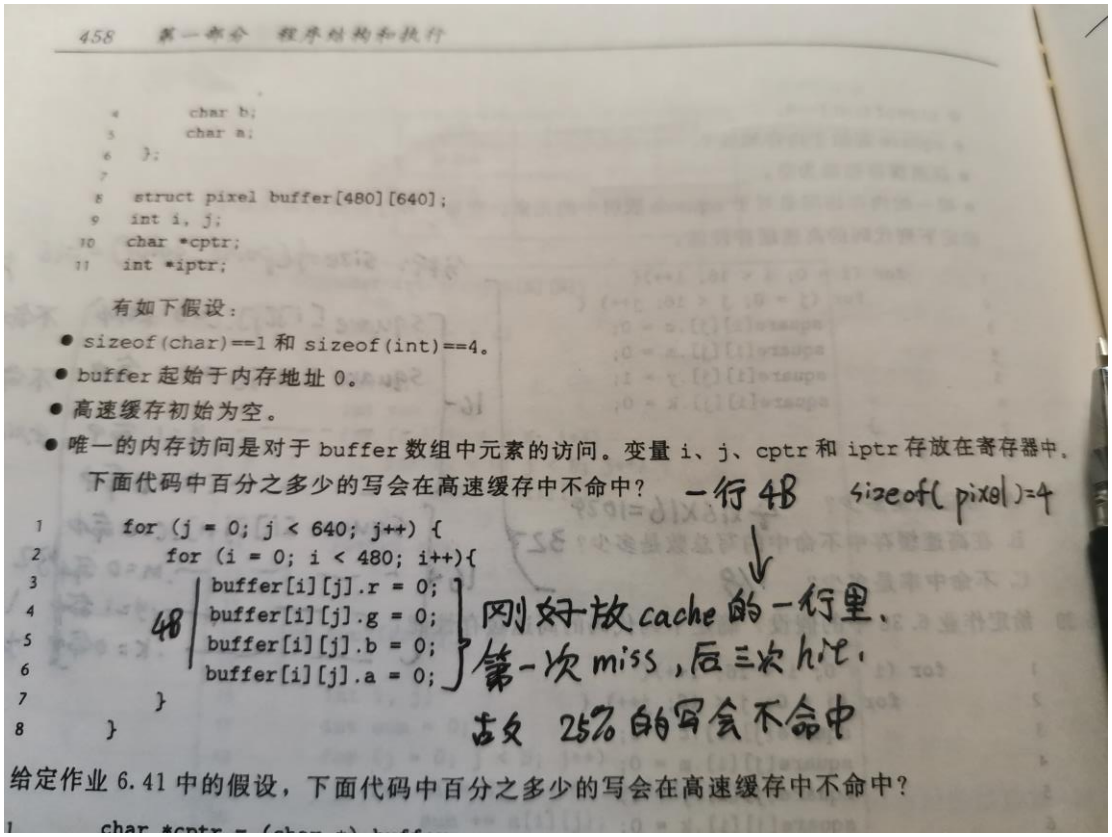
B. 在高速缓存中不命中的写总数是多少？ 32

C. 不命中率是多少？ 1/8

确定作业 6.38 中的假设，确定下列代码的高速缓存性能：

```
for (i = 0; i < 16; i++){
    for (j = 0; j < 16; j++) {
        square[j][i].c = 0;
        square[j][i].m = 0;
        square[j][i].y = 1;
        square[j][i].k = 0;
```

6.41：



458 第一部分 程序结构和执行

```
4       char b;
5       char a;
6   };
7
8   struct pixel buffer[480][640];
9   int i, j;
10  char *cptr;
11  int *iptr;
```

有如下假设：

- sizeof(char)==1 和 sizeof(int)==4。
- buffer 起始于内存地址 0。
- 高速缓存初始为空。
- 唯一的内存访问是对于 buffer 数组中元素的访问。变量 i、j、cptr 和 iptr 存放在寄存器中。

下面代码中百分之多少的写会在高速缓存中不命中？ 一行 4B sizeof(pixel)=4

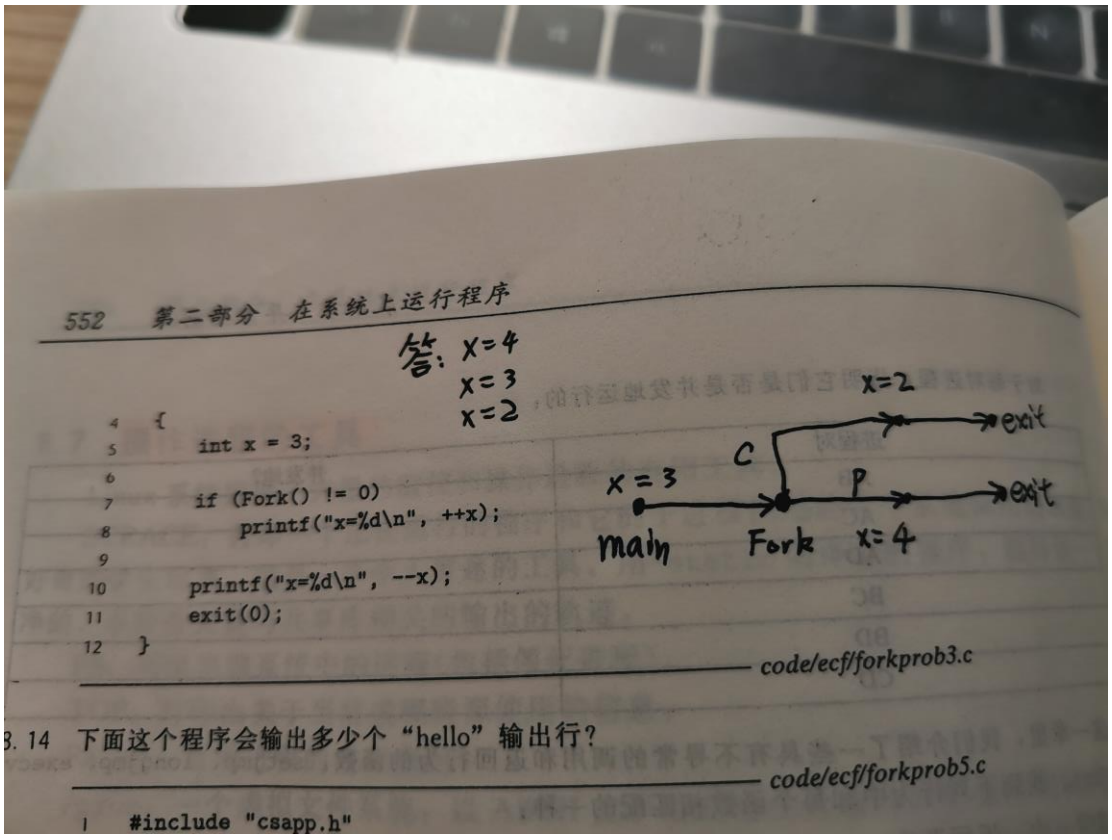
```
1   for (j = 0; j < 640; j++) {
2       for (i = 0; i < 480; i++){
3           buffer[i][j].r = 0;
4           buffer[i][j].g = 0;
5           buffer[i][j].b = 0;
6           buffer[i][j].a = 0;
7       }
8   }
```

48 刚好放cache的一行里，第一次miss，后三次hit，故 25% 的写会不命中

给定作业 6.41 中的假设，下面代码中百分之多少的写会在高速缓存中不命中？

```
1   char *cptr = (char *) buffer;
```

8.13：



552 第二部分 在系统上运行程序

答：x=4
x=3
x=2

```
4   {
5       int x = 3;
6
7       if (Fork() != 0)
8           printf("x=%d\n", ++x);
9
10      printf("x=%d\n", --x);
11      exit(0);
12  }
```

code/ecf/forkprob3.c

8.14 下面这个程序会输出多少个“hello”输出行？

code/ecf/forkprob5.c

```
1   #include "csapp.h"
```

8.15：

```
}

int main()
{
    doit();
    printf("hello\n");
    exit(0);
}
```

code/ecf/forkprob5.c

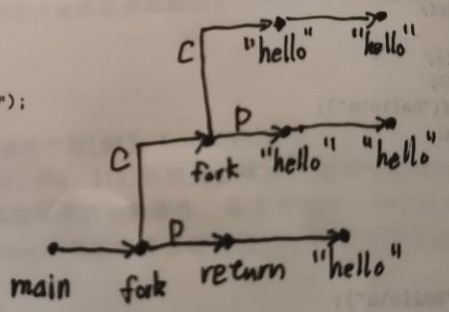
• 8.15 下面这个程序会输出多少个“hello”输出行？ 答:5行

code/ecf/forkprob6.c

```
#include "csapp.h"

void doit()
{
    if (Fork() == 0) {
        Fork();
        printf("hello\n");
        return;
    }
    return;
}

int main()
{
    doit();
    printf("hello\n");
    exit(0);
}
```



code/ecf/forkprob6.c

• 8.16 下面这个程序的输出是什么？

8.18：

```
    exit(0);
}
```

code/ecf/forkprob7.c

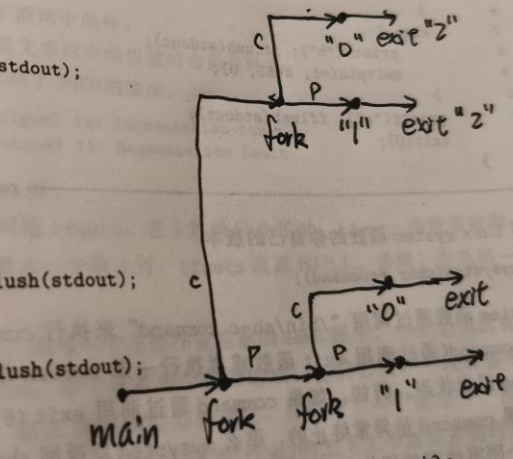
列举练习题 8.4 中程序所有可能的输出。
考虑下面的程序 答:A C E 可能

```
#include "csapp.h"

void end(void)
{
    printf("2"); fflush(stdout);
}

int main()
{
    if (Fork() == 0)
        atexit(end);
    if (Fork() == 0) {
        printf("0"); fflush(stdout);
    }
    else {
        printf("1"); fflush(stdout);
    }
    exit(0);
}
```

code/ecf/forkprob2.c



code/ecf/forkprob2.c

判断下面哪个输出是可能的。注意：atexit 函数以一个指向函数的指针为输入，并将它添
调用时，会调用该列表中的函数。
E. 100