北京大学信息科学技术学院考试试卷

考试科目：计算机系统导论 **姓名：**_____ **学号：**_____

考试时间： 2017 年 1 月 2 日 **小班教师：**_____

题号	一	二	三	四	五	六	七	八	**总分**
分数									
阅卷人									

北京大学考场纪律

1、考生进入考场后，按照监考老师安排隔位就座，将学生证放在桌面上。无学生证者不能参加考试；迟到超过 15 分钟不得入场。在考试开始 30 分钟后方可交卷出场。

2、除必要的文具和主考教师允许的工具书、参考书、计算器以外，其它所有物品（包括空白纸张、手机、或有存储、编程、查询功能的电子用品等）不得带入座位，已经带入考场的必须放在监考人员指定的位置。

3、考试使用的试题、答卷、草稿纸由监考人员统一发放，考试结束时收回，一律不准带出考场。若有试题印制问题请向监考教师提出，不得向其他考生询问。提前答完试卷，应举手示意请监考人员收卷后方可离开；交卷后不得在考场内逗留或在附近高声交谈。未交卷擅自离开考场，不得重新进入考场答卷。考试结束时间到，考生立即停止答卷，在座位上等待监考人员收卷清点后，方可离场。

4、考生要严格遵守考场规则，在规定时间内独立完成答卷。不准交头接耳，不准偷看、夹带、抄袭或者有意让他人抄袭答题内容，不准接传答案或者试卷等。凡有违纪作弊者，一经发现，当场取消其考试资格，并根据《北京大学本科考试工作与学术规范条例》及相关规定严肃处理。

5、考生须确认自己填写的个人信息真实、准确，并承担信息填写错误带来的一切责任与后果。

学校倡议所有考生以北京大学学生的荣誉与诚信答卷，共同维护北京大学的学术声誉。

以下为试题和答题纸，共 22 页。

装订线内

不要答题

得分

第一题 单项选择题（每小题 1 分，共 20 分）

注：选择题的回答填写在下表中。

题号	1	2	3	4	5	6	7	8	9	10
回答										
题号	11	12	13	14	15	16	17	18	19	20
回答										

1. C 语言中的 int 和 unsigned 类型的常数进行比较时，下列表达式及描述正确的是：

（注：位宽为 32 位，T_{MIN}=-2,147,483,648，T_{MAX}=2,147,483,647）

A. 0 == 0U，按有符号数进行比较

B. 2147483647U > -2147483647-1，按无符号数进行比较

C. (unsigned)-1 < -2，按无符号数进行比较

D. 2147483647 > (int)2147483648U，按有符号数进行比较

2. 现有一个二进制浮点的表示规则，其中 E 为指数部分，3 比特，且 bias 为 3；M 为小数部分，5 比特，采用二进制补码表示形式，且取值(½ ≤ |M|<1)；s 是浮点的符号位；该形式包含一个值为 1 的隐藏位。

s	E	M

如果用该形式表示+5_{10}，s、E 和 M 分别是：

A. 0, 100, 01100

B. 0, 101, 00100

C. 0, 110, 11010

D. 0, 111, 10101

3. 缓冲区溢出会带来程序风险，下列避免方法中错误的是：

A. 在栈中存放特殊字段用于检测是否发生缓冲区溢出

B. 避免使用有风险的库函数，如 gets 等

C. 随机设置栈的偏移地址

D. 分配尽可能大的缓冲区数组

4. 现有四级指令流水线，分别完成取指、取数、运算、传送结果 4 步操作。若完成上述操作的时间依次为 9ns、10ns、6ns、8ns，则流水线的操作周期应设计为 _____ ns。

A. 6　　B. 8　　C. 9　　D. 10

5. 对于下面这段程序，哪个描述是正确的：_____

```
void upper(char *s)
{
    for(int i=0; i<strlen(s); i++)
        if(s[i]>='a' && s[i]<='z')
            s[i] +=('A'-'a');
}
```

A. 假设 s 的长度为 N，该程序的时间复杂度为 O(NlogN)

B. 将"for(int i=0; i<strlen(s); i++){}"修改为:
"int len = strlen(s); for(int i=0; i<len; i++){}"
可以使程序运行时间与 S 的长度线性相关

C. B 选项中的修改策略不影响程序的空间局部性与时间局部性

D. B 选项中的修改策略仅影响程序的空间局部性，不影响时间局部性

6. 某机器内存为 8 位地址，cache 设计参数为 E=2,t=2,s=2,b=4，cache 块大小为16字节。cache 为空的初始状态下,数据访问的地址序列为 0->4->34->162->128->192->2(以字节为单位),请问一共发生多少次 cache 命中?

A. 0　　B. 1　　C. 2　　D. 3

7. C 源文件 f1.c 和 f2.c 的代码分别如下所示，编译链接生成可执行文件后执行，输出结果为:

A. 100　　B. 200
C. 201　　D. 链接错误

```
// f1.c
#include <stdio.h>
static int var = 100;
int main(void)
{
    extern int var ;
    extern void f() ;
    f() ;
    printf("%d\n", var) ;
    return 0;
}
```

```
//f2.c
int var = 200;

void f()
{
    var++;
}
```

8. C 源文件 m1.c 和 m2.c 的代码分别如下所示，编译链接生成可执行文件后执行，结果最可能为:

```
// m1.c
#include <stdio.h>

int a1 ;
int a2 = 2 ;
extern int a4 ;

void hello()
{
    printf("%p ", &a1);
    printf("%p ", &a2);
    printf("%p\n", &a4);
}
```

```
//m2.c
int a4 = 10 ;

int main()
{
    extern void hello() ;

    hello() ;
    return 0 ;
}
```

```
$ gcc -o a.out m2.c m1.c ; ./a.out
0x1083020
```

A. 0x1083018, 0x108301c　　B. 0x1083028, 0x1083024
C. 0x1083024, 0x1083028　　D. 0x108301c, 0x1083018

9. 对于以下一段代码，可能的输出为：

```
int count = 0;
int pid = fork();
if (pid == 0){
   printf("count = %d\n",--count);
}
else{
   printf("count = %d\n",++count);
}
printf("count = %d\n",++count);
```

A.1 2 -1 0　　B.0 0 -1 1
C.1 -1 0 0　　D.0 -1 1 2

10. 下列哪一事件不会导致信号被发送到进程？
    A. 新连接到达监听端口
    B. 进程访问非法地址
    C. 除零
    D. 上述情况都不对

11. 在 C 语言中实现 Mark-and-Sweep 算法时，可以基于以下哪个假设：（宿主机为 32 位机器）
    A. 所有指针指向一个块的起始地址
    B. 所有指针数据都是 4 字节对齐
    C. 只需要扫描数据类型为指针的堆中的数据空间
    D. 只需要扫描所有长度为 4 字节的堆中的数据空间

12. 在 Core i7 中，以下哪个页表项属于 4 级页表项，不属于 1 级页表项：
    A. G 位（Global Bit）
    B. D 位（Dirty Bit）
    C. XD 位（Disable or enable instruction fetch）
    D. U/S 位（User or supervisor mode access permission）

13. 在 Core i7 中，关于虚拟地址和物理地址的说法，不正确的是：
    A. VPO = CI + CO
    B. PPN = TLBT + TLBI
    C. VPN1 = VPN2 = VPN3 = VPN4
    D. TLBT + TLBI = VPN

14. 考虑以下代码，假设 result.txt 中的初始内容为“666666”

```
char *str1 = "6666";
char *str2 = "2333";
char *str3 = "hhhh";
int fd1, fd2, fd3, i;

fd1 = open("result.txt", O_RDWR);
fd2 = open("result.txt", O_RDWR);
dup2(fd1, fd2);

for (i = 0; i < 5; ++i) {
    fd3 = open("result.txt", O_RDWR);
    write(fd1, str1, 4);
    write(fd2, str2, 4);
    write(fd3, str3, 4);
    close(fd3);
}

close(fd1); close(fd2);
```

假设所有系统调用均成功，则这段代码执行结束后，result.txt 的内容中有（）个“6”

A. 6
B. 16
C. 20
D. 22

15. 关于 IO 操作，以下说法中正确的是（）

A. 由于 RIO 包的健壮性，所以 RIO 中的函数都可以交叉调用
B. 成功调用 open 函数后，返回一个不小于 3 的文件描述符
C. 调用 Unix I/O 开销较大，标准 I/O 库使用缓冲区来加快 I/O 的速度
D. 和描述符表一样，每个进程拥有独立的打开文件表

16. 关于 IP，以下说法正确的是：

A. IP 协议提供了可信赖的主机与主机之间的数据包传输能力
B. IP 地址的长度固定为 32 位
C. 一个域名可以映射到多个 IP，但一个 IP 不能被多个域名映射
D. 一个域名可以不映射到任何 IP

17. 关于以下说法，以下说法正确的是：

A. HTTP 协议规定服务器端使用 80 端口提供服务
B. 使用 TCP 来实现数据传输一定是可靠的
C. Internet 上的两台的主机要通信必须先建立端到端连接
D. 在 Linux 中只能通过 Socket 接口进行网络编程

18. 假设有一个 HTTPS（基于 HTTP 的一种安全的应用层协议）客户端程序想要通过一个 URL 连接一个电子商务网络服务器获取一个文件，并且这个服务器 URL 的 IP 地址是已知的，以下哪种协议是一定不需要的？

A. HTTP
B. TCP
C. DNS
D. SSL/TLS

19. 下列关于“基于进程的并发服务器”的叙述中，哪一个是错误的？

A. 父子进程共享文件表
B. 父子进程共享文件描述符
C. 父子进程不共享用户地址空间
D. 进程控制和进程间通信开销大

20. 请阅读如下代码：

```
sem_t s;

int main()
{
    int i;
    pthread_t tids[3];
    sem_inti(&s, 0, 1);
    for (i=0; i <3; i++) {
       pthread_create(&tids[i], NULL, justdoit, NULL);
    }
    for (i=0; i <3; i++) {
       pthread_join(&tids[i], NULL);
    }
    return 0;
}
int j=0;

void *justdoit(void *arg)
{
    P(&s);
    j = j + 1;
    V(&s);
    printf("%d\n", j);
}
```

下列哪一个输出结果是不可能的？

A. （1，3，2）
B. （2，3，2）
C. （3，3，2）
D. （2，1，2）

得分

第二题（12 分）

这是一个完整 C 函数 myfunction 经编译器优化后生成的 32 位汇编语言代码：

```
myfunction:
    pushl   %ecx
    movl 0x10(%esp), %ecx
    movl 8(%esp), %edx
    pushl %ebx
    movl 0x18(%esp), %ebx
    pushl %esi
    movl 0x14(%esp), %esi
    pushl %ebp
    xorl %eax, %eax
    pushl %edi
    jmp .L2
    leal (%esp), %esp
.L2:
    movl 8(%esi), %edi
    imull 4(%edx,%eax,8), %edi
    movl (%esi), %ebp
    imull (%edx,%eax,8), %ebp
    addl %ebp, %edi
    movl %edi, (%ecx)
    movl 0xc(%esi), %edi
    imull 4(%edx,%eax,8), %edi
    movl 4(%esi), %ebp
    imull (%edx,%eax,8), %ebp
    addl %ebp, %edi
    movl (%ecx), %ebp
    addl %edi, %ebp
    addl %ebp, %ebx
    movl %edi, 4(%ecx)
    incl %eax
    addl $8, %ecx
    cmpl $2, %eax
    jl .L2
    popl %edi
    popl %ebp
    popl %esi
    movl %ebx, %eax
    popl %ebx
    popl %ecx
    ret
```

提示：32 位汇编使用栈来传递所有参数，参数压栈顺序为从右至左。

1．请根据上述汇编代码，根据提示，补全 myfunction 的代码。注意，每行一条语句；不能定义新的变量。

```
int myfunction(int a[], int b[], int c[], int d) {
    int i;
    for (i = 0; i < _____; i++) {

        _________ = ____________________________________;

        _________ = ____________________________________;

        _________ = ____________________________________;
    }
    return _____;
}
```

2．请给出以下程序的输出结果。

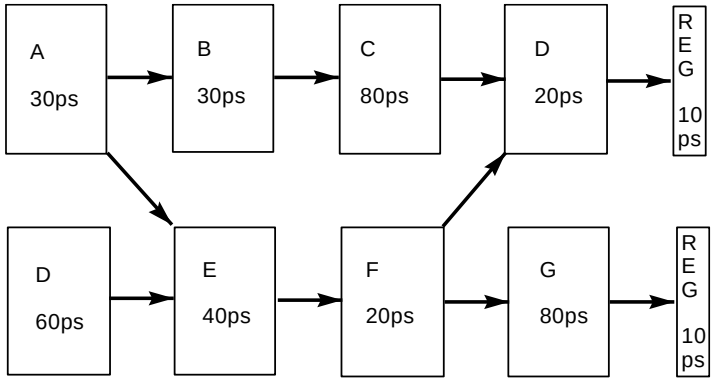
```
int main( )
{
    int a[5] = {1, 2, 3, 4, 5};
    int b[5] = {9, 8, 5, 6, 4};
    int c[5] = {7, 9, 8, 10, 11};
    int d = 5;
    int ret = myfunction(a, b, c, d);
    printf("%d\n", ret);
    return 0;
}
```

答：

得分

第三题（12 分）

如图所示，每个模块表示一个单独的组合逻辑单元，每个单元的延迟以及数据依赖关系已在图中标出。通过在两个单元间添加寄存器的方式，可以对该数据通路进行流水化改造。假设每个寄存器的延迟为 10ps。



1）如果改造为一个二级流水线，为获得最大的吞吐率，该寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。结果可以是分数形式也可以是小数形式。

2）如果改造为一个三级流水线，为获得最大的吞吐率，寄存器应在哪里插入？请计算该流水线的吞吐率，并说明计算过程。结果可以是分数形式也可以是小数形式。

得分

第四题（10 分）

在 x86_64 环境下，考虑如下 4 个文件（main.c, value.c, f1.c, f2.c）：

```
/* main.c */
#include <stdio.h>

extern void f_void_void();
extern int f_int_void();
void *f;

int main()
{
    int a = 1, b, c;

    f = (void *)f_void_void;

    ((void (*)(int))f)(a);
    b = ((int (*)())f)();
    printf("b = %d\n", b);

    f = (void *)f_int_void;

    ((void (*)())f)(a);
    c = ((int (*)(int))f)(a);
printf("c = %d\n", c);

    return 0;
}

/* value.c */
int BIG;

/* f1.c */
#include <stdio.h>

extern int BIG;
int small = 1;

void f_void_void() {
    small += 1;
    BIG += 1;
    printf("small = %d, BIG = %d\n", small, BIG);
}
```

```c
/* f2.c*/
#include <stdio.h>

extern int BIG;
static int small;

int f_int_void() {
   small += 1;
   BIG += 1;
   printf("small = %d, BIG = %d\n", small, BIG);
   return small + 1;
}
```

使用命令

`gcc -o main main.c f1.c f2.c value.c`

编译这四个文件。

使用

`./main`

运行编译好的程序。

1. 对于程序中的相应符号，请给出它的属性（局部或全局，强符号或弱符号），不确定的请画 X。

源文件	符号名	局部或全局？	强符号或弱符号？
main.c	f		
value.c	BIG		
f1.c	small		
f2.c	small		

2. 请补全程序运行的输出，不确定的请画 X。

```
small = _____, BIG = _____
small = _____, BIG = _____
b = _____
small = _____, BIG = _____
small = _____, BIG = _____
c = _____
```

得分

第五题（10 分）

Part I

请阅读以下程序，然后回答问题。假设程序中的函数调用都可以正确执行，并默认 printf 执行完会调用 fflush。

```
int main() {
    int cnt=1;
    int pid_1,pid_2;
    pid_1=fork();
    if(pid_1==0)    {
        pid_2=fork();
        if(pid_2!=0)    {
            wait(pid_2,NULL,0);
            printf("B");
        }
        printf("F");
        exit(0);
    }
    else {
        A
        B
        wait(pid_1,NULL,0);
        pid_2=fork();
        if(pid_2==0)    {
            printf("D");
            cnt-=1;
        }
        if(cnt==0)
             printf("E");
        else
            printf("G");
        exit(0);
    }
}
```

（1）如果程序中的 A、B 位置的代码为空，列出所有可能的输出结果：

（2）如果程序中的 A、B 位置的代码为：

A: printf("C");

B: exit(0);

列出所有可能的输出结果：

Part II

请阅读以下程序，然后回答问题（假设程序中的函数调用都可以正确执行，且每条语句都是原子动作）：

```
pid_t pid;
int even = 0;
int counter1 = 0;
int counter2 = 1;
void handler1(int sig) {
    if (even % 2 == 0) {
        printf("%d\n", counter1);
        counter1 = ____A_____ ;
        } else {
            printf("%d\n", counter2);
            counter2 = ____B____ ;
        }
        even = even+___C_____ ;
    }
void handler2(int sig) {
        if (_____D_____) {
        counter1 = even*even;
        } else {
        counter2 = even*even;
        }
}
int main() {
    signal(SIGUSR1, handler1);
    signal(SIGUSR2, handler2);
```

```
if ((pid = fork()) == 0) {
    while (1) {};
}
    while (even < 30) {
        kill(pid,  E  );
        sleep(1);
        kill(pid,  F  );
        sleep(1);
        even = even+  G  ;
    }
    kill(pid, SIGKILL);
    exit(0);
}
```

（1）完成程序，使得程序在输出的数字为以下 Q 队列的前 30 项，Q 队列定义如下：

$$Q_0 = 0,\ Q_1 = 1,\ Q_{n+2} = \begin{cases} Q_n + 1, & \text{n\%2} = 0 \\ Q_n \times 2, & \text{n\%2} \neq 0 \end{cases} \quad (n = 0,1,2,3,\ldots)$$

（注：若某个位置中的程序内容，对本次程序执行结果没有影响，请在相应位置填写“无关”）

A: _____

B: _____

C: _____

D: _____

E: _____

F: _____

G: _____

得分

第六题（12 分）

程序 i16.c 如下：

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/wait.h>
#include "csapp.h"

int main(int argc, char **argv)
{
    int fd1, fd2;
    char *cs;
    struct stat stat;
    int sz, i;

    fd1 = Open(argv[1], O_RDWR, S_IRUSR | S_IWUSR);
    Fstat(fd1, &stat);
    sz = stat.st_size;
    fd2 = Open(argv[2], O_RDWR | O_TRUNC | O_CREAT, \
               S_IRUSR | S_IWUSR);

    cs = (char *)Mmap(NULL, sz, PROT_READ | PROT_WRITE, \
            _____X_____, fd1, 0);
    if (Fork()) {
        Wait(&i);
        Write(fd2, cs, sz);
    } else {
        for (i = 0; i < sz; i++) {
            if (islower(cs[i]))
                cs[i] = (char) toupper(cs[i]);
            else if (isupper(cs[i]))
                cs[i] = (char) tolower(cs[i]);
        }
        _____Y_____;
    }
    Munmap(cs, sz);
    Close(fd1);
    Close(fd2);
}
```

1、假定 t1 内容为 ICSexam，根据 X/Y 处的内容，判断执行./i16 t1 t2 之后的结果：（每小题均只有一个正确答案）

1.1、X 为 MAP_PRIVATE，Y 为空时，结果为（　　）

A. t1 为 ICSexam，t2 为 ICSexam

B. t1 为 ICSexam，t2 为 icsEXAM

C. t1 为 icsEXAM，t2 为 icsEXAM

D. t1 为 icsEXAM，t2 为 ICSexam

1.2、X 为 MAP_SHARED 时，Y 为空时，结果为（　　）

A. t1 为 ICSexam，t2 为 ICSexam

B. t1 为 ICSexam，t2 为 icsEXAM

C. t1 为 icsEXAM，t2 为 icsEXAM

D. t1 为 icsEXAM，t2 为 ICSexam

1.3、X 处的内容为 MAP_PRIVATE，Y 为 Write(fd2, cs, sz)时，t2 的结果为（　　）

A. t2 为 ICSexam

B. t2 为 icsEXAM

C. t2 为 icsEXAMICSexam

D. t2 为 icsEXAMicsEXAM

1.4、X 处的内容为 MAP_SHARED，Y 为 Write(fd2, cs, sz)时，t2 的结果为（　　）

A. t2 为 ICSexam

B. t2 为 icsEXAM

C. t2 为 icsEXAMICSexam

D. t2 为 icsEXAMicsEXAM

2、X 为 MAP_SHARED，当子进程运行到 Y 处时，内存映射的内容如下：（只保留了/proc/pid/maps 中的三部分内容：分别为 address/perms/pathname）

```
00400000-00405000 r-xp         /home/tao/i16/i16
00604000-00605000 r--p         /home/tao/i16/i16
00605000-00606000 __A___       /home/tao/i16/i16
7f5df1e61000-7f5df2020000 r-xp  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2020000-7f5df2220000 ---p  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2220000-7f5df2224000 r--p  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2224000-7f5df2226000 rw-p  /lib/x86_64-linux-gnu/libc-2.23.so
7f5df2226000-7f5df222a000 rw-p
7f5df222a000-7f5df2250000 r-xp  /lib/x86_64-linux-gnu/____B_____
7f5df243f000-7f5df2442000 rw-p
7f5df244c000-7f5df244d000 rw-s  _______C_____
7f5df244d000-7f5df244f000 rw-p
7f5df244f000-7f5df2450000 r--p  /lib/x86_64-linux-gnu/____B_____
7f5df2450000-7f5df2451000 rw-p  /lib/x86_64-linux-gnu/____B_____
7f5df2451000-7f5df2452000 rw-p
7ffd4e35f000-7ffd4e380000 rw-p  [____D_____]
7ffd4e3b0000-7ffd4e3b2000 r--p  [vvar]
7ffd4e3b2000-7ffd4e3b4000 r-xp  [vdso]
ffffffffff600000-ffffffffff601000 r-xp  [vsyscall]
```

根据程序执行情况，填充空白处：

A：_____

B：_____

C：_____

D：_____

3、X 为 MAP_SHARED，当子进程运行到 Y 处时：（每小题均只有一个正确答案）

3.1、关于页表的描述，正确的是：（　　）

A．由于 i16 本身的代码和数据只占用了低 32 位空间，所以这部分空间只需使用 2 级页表

B．所有虚拟地址空间都是 48 位地址空间，都需要使用完整的 4 级页表

C．当页表项无效时（P 位为 0），MMU 不会使用到其它位的内容

D．除读写权限外，需要在页表项中为 COW 机制提供专门的支持

3.2、如果子进程在 Y 处访问 7f5df2224 这一页，可能发生以下异常：（　）

A. Page Fault 或 General Protection Fault

B. General Protection Fault 或 Segmentation Fault

C. Page Fault 或 Segmentation Fault

D. Page Fault 或 General Protection Fault 或 Segmentation Fault

3.3、假设 TLB 有 64 个表项，4 路组相联，当访问 7f5df2224 这一页时，所对应的 TLBT 应为：（　）

A. 0111 1111 0101 1101 1111 0010 0010 0010

B. 0111 1111 0101 1101 1111 0010 0010 0010 01

C. 1111 0101 1101 1111 0010 0010 0010 0100

D. 11 1111 0101 1101 1111 0010 0010 0010 0100

3.4、当发生 TLB 命中时，则意味着：（　）

A. 相应的页表项，在 L1 Cache 里

B. 要访问的数据内容，在 L1 Cache 里

C. 如果要访问的数据内容已在 L1 Cache 中，则该 Cache Line 的权限位中应具有相应的读写执行权限

D. 以上都不对

得分

第七题（12 分）

下面是一个基于进程的并发 echo server 的主要代码(省略了与本题无关的部分)，每段代码都注释了相应的功能，请补充空缺的代码（**可能是 0 行或多行**)。请确保任何由你打开的文件描述符都被显式地关闭。

```
int open_listenfd(char *port) {
    struct addrinfo hints, *listp, *p;
    int listenfd, optval = 1;

    /* Get a list of potential server addresses */
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_socktype = SOCK_STREAM;
    hints.ai_flags = AI_PASSIVE | AI_ADDRCONFIG;
    hints.ai_flags |= AI_NUMERICSERV;
    Getaddrinfo(NULL, port, &hints, &listp);

    /* Walk the list for one that we can bind to */
    for (p = listp; p; p = p->ai_next) {
        /* Create a socket descriptor. If failed, try the next */
        if ((listenfd = socket(p->ai_family,
            p->ai_socktype, p->ai_protocol)) < 0)
            (1)
        /* Eliminates error from bind */
        Setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR,
            (const void *)&optval, sizeof(int));
        /* Bind the descriptor to the address */
        if (bind(listenfd, p->ai_addr, p->ai_addrlen) == 0)
            (2) else /* Success, use the current one */
            (3)      /* Bind failed, try the next */
    }
    (4)
    if (!p) return -1; /* No address worked */

    /* Make it a listening socket*/
    if (listen(listenfd, LISTENQ) < 0) {
        (5)
        return -1;
    }
    return listenfd;
}
```

```
void echo(int fd); /* service a client via fd */
void handler(int sig) {
    while (waitpid(-1, 0, WNOHANG) > 0);
    return;
}
int main(int argc, char **argv) {
    int listenfd, connfd;
    socklen_t clientlen;
    struct sockaddr_storage clientaddr;

    Signal(    (10)    , handler);
    listenfd = Open_listenfd(argv[1]);
    while (1) {
        clientlen = sizeof(struct sockaddr_storage)
        __(6)__ = Accept(__(7)__, (SA *)&clientaddr, &clientlen);
        if (Fork() == 0) {
            /* Child services client and close fd */
            __(8)__
            exit(0); /* Child exits */
        }
        __(9)__
    }
}
```

答：

(1) _____

(2) _____

(3) _____

(4) _____

(5) _____

(6) _____

(7) _____

(8) _____

(9) _____

(10) _____

得分

第八题（12 分）

1.请阅读下列代码 badcnt.c：

```
/* Global shared variable */
volatile long cnt = 0; /* Counter */
int main(int argc, char **argv)
{
    long niters;
    pthread_t tid1, tid2;
    niters = atoi(argv[1]);
    Pthread_create(&tid1, NULL, thread, &niters);
    Pthread_create(&tid2, NULL, thread, &niters);
    Pthread_join(tid1, NULL);
    Pthread_join(tid2, NULL);
    /* Check result */
    if (cnt != (2 * niters))
        printf("BOOM! cnt=%ld\n", cnt);
    else
        printf("OK cnt=%ld\n", cnt);
    exit(0);
}

/* Thread routine */
void *thread(void *vargp)
{
    long i, niters = *((long *)vargp);

    for (i = 0; i < niters; i++)
        cnt++;

    return NULL;
}
```

运行后可能产生如下结果：

```
linux> ./badcnt 10000
OK cnt=20000
```

或者：

```
linux> ./badcnt 10000
BOOM! cnt=13051
linux>
```

为什么会产生不同的结果？请分析产生不同结果的原因。

2.某辆公交车的司机和售票员为保证乘客的安全，需要密切配合、协调工作。司机和售票员的工作流程如下所示。请编写程序，用 P、V 操作来实现司机与售票员之间的同步。

司机进程：

```
while(1) {
    ①
    启动车辆;
    ②
    正常行驶;
    ③
    到站停车;
    ④
}
```

售票员进程：

```
while(1) {
    ⑤
    关门;
    ⑥
    报站名或维持秩序;
    ⑦
    到站开门;
    ⑧
}
```

（1）请设计若干信号量，给出每一个信号量的作用和初值。

（2）请将信号量对应的 PV 操作填写在代码中适当位置。

注：每一标号处可以不填入语句（请标记成 X），或填入一条或多条语句。

标号	对应的操作
①	
②	
③	
④	
⑤	
⑥	
⑦	
⑧	