

2015-2016 学年第二学期期末考试 A 卷

一、单选题(每小题 3 分, 共 30 分)

1、在_____,指令#include<stdio.h>会被替换为文件 stdio.h 的内容()。

- A、编译代码时 B、运行程序时 C、编译预处理时 D、程序连接时

2、把数组 int a[10][20]传入函数 aver 求平均值, 不正确的代码段是()

- | | |
|---|---|
| <p>A、int a[10][20]={1,2,3,4,5};
aver(a,200);
float aver(int *a,int size)
{ /*...*/ }</p> | <p>B、int a[10][20]={1,2,3,4,5};
aver(a,10);
float aver(int (*a)[20],int rows)
{ /*...*/ }</p> |
| <p>C、int a[10][20]={1,2,3,4,5};
aver(*a,200);
float aver(int *a,int size)
{ /*...*/ }</p> | <p>D、int a[10][20]={1,2,3,4,5};
aver(a,10);
float aver(int a[][20],int rows)
{ /*...*/ }</p> |

3、关于递归函数调用的说法, 错误的是()。

- A、每次调用函数时分配参数和局部变量的存储空间, 退出函数时释放;
B、随着递归函数的层层深入, 存储空间的一端逐渐增加, 然后随着函数调用的层层返回, 存储空间的一端又逐渐缩短;
C、递归函数存在着可用堆栈空间过度使用的危险;
D、递归调用的层数可以是无限制的。

4、已知 sort()函数原型为: void sort(int a[],int size);, 并且另有 fun()函数, 它的第一个参数为整型指针, 第二个参数可以接受函数 sort 的入口地址, 则函数 fun 的函数原型为()。

- A、void fun(int *p, void(*p)(int a[],int size));
B、void fun(int *p, void *q);
C、void fun(int *p, void(*q)(int a[],int size));
D、void fun(int *p, void(*p)(int a[size]));

5、定义宏 MAX, 求两个数的最大值, 最合适的是()。

- A、#define MAX(a,b) (a>b)?a:b
B、#define MAX (a,b) (a>b)?a:b
C、#define MAX(a,b) ((a)>(b)?(a):(b))
D、#define MAX (a,b) (((a)>(b))?(a):(b));

6、在下面的代码中, p2 是一个()。

```
typedef int *ptr;
ptr p1,p2;
```

- A、整数 B、指向整数的指针 C、代码有语法错误 D、以上都不对

7、假设高考考生成绩与简单信息的结构类型定义如下:

```
struct scoretype{
int Math,Chinese,English,Integrated>Total;
};
struct student{
char name[20];
```

```
unsigned long num; //准考证号码
```

```
unsigned long rank; //排名序号
```

```
struct scoretype score;
```

```
    }s[100],*p;
```

则“数学 Math 成绩超过 140 分的同学总分 Total 额外再加 5 分”的正确写法是()。

A、if(p->score->Math>140) p->score->Total+=5;

B、if(*p->score.Math>140) *p->score.Total+=5;

C、if(*p.score.Math>140) *p.score.Total+=5;

D、if(p->score.Math>140) (*p).score.Total+=5;

8、假设单链表的结点结构定义如下:

```
struct node{
```

```
int data; struct node *next;
```

```
}*p,*n;
```

并且假设当前 p 指向单链表中的某内部结点(既非首结点也非尾结点),删除该结点的正确语句序列是()。

A、n=p->next; p->next=n->next; free(n);

B、n=p->next; p->data=n->data; free(n);

C、n=p->next; p->next=n->next; p->data=n->data; free(p+1);

D、n=p->next; p->data=n->data; p->next=p->next->next; free(n);

9、在最坏情况下,选择排序算法时间复杂度是()。

A、O(N)

B、O(N²)

C、O(2N)

D、O(N³)

10、若归并非递归排序算法思想为:

第一趟归并:以间隔为 1 的进行归并,也就是说,第一个元素跟第二个进行归并。第三个与第四个进行归并;

第二趟归并:以间隔为 2 的进行归并, [(1) (2)]与[(3) (4)]进行归并, [(5) (6)]与[(7) (8)]进行归并;

第三趟归并:以间隔为 2*2 的进行归并;

同理,直到 2k 超过数组长度为止。

则按照上述算法,对 4 3 7 8 0 9 2 1 5 6 进行排序。第二趟归并后的结果是()。

A、3 4 7 8 0 1 2 9 5 6

B、0 1 2 3 4 7 8 9 5 6

C、3 4 7 0 8 9 2 1 5 6

D、3 4 0 7 8 9 2 1 5 6

二、改错题(每小题 3 分,共 18 分)

1、找出并改正以下结构定义中的错误。

```
struct StudentListNode{
    int num,score;
    StudentListNode * next;
};
```

2、以下的递归函数将任意十进制整数 num 转换成 base 进制数($1 < \text{base} < 10$)输出,找出错误并修改正确。

```
void change(int num, int base){
if(num/base>0) {
printf("%d", num % base);
change(num / base, base);
}
else printf("%d", num);
}
```

3、以下程序使用了条件编译,本次运行想输出“BBBB”,找出错误并修改正确。

```
#include <stdio.h>
#define DEBUG 0
int main(void ){
    #ifdef DEBUG
        printf("AAAA\n");
    #else
        printf("BBBB\n");
    #endif
    return 0;
}
```

4、函数 Find 在数组 x 中查找值为 v 的元素,并返回元素的下标。如果满足要求的元素不存在,那么返回-1。假设参数 x 的元素按照升序排序。找出错误并修改正确。

```
int Find(float x[], int n, float v){
    int m = n/2; /*数组中间下标*/
if(n<1 ) return (-1);
if(x[m]<v ) return Find(x+m+1,n,v);
else
if( x[m]>v ) return Find(x,m-1,v);
else return m;
}
```

5、以下程序把结构变量 a 赋值给 b，再输出结构变量 b，结果输出的 b.name 是乱码，而 b.score 是正常的。找出错误并修改正确。

```
typedef struct student {
    char *name;
    int score;
}ST;
void main() {
    ST a, b;
    a.name = malloc(100);
    scanf("%s %d", a.name, &a.score);
    b=a;
    free(a.name);
    printf("%s %d\n", b.name, b.score);
}
```

6、函数 buildlist()用来创建一个双向链表 main()在调用 buildlist()后，按逆序输出链表中的各个节点:4 3 2 1 0。已知函数 buildlist()中有一个语句有逻辑错误，找出该语句并修改正确。

```
#include <stdio.h>
typedef struct list {
    int data;
    struct list *prev;
    struct list *next;
}LIST;
LIST * buildlist(int n) {
    int i;
    LIST *phead = NULL, *pfpre = NULL, *p;
    for(i=0; i<n; i++){
        p= malloc(sizeof(LIST));
        p->data = i;
        p->next = NULL;
        p->prev = pfpre;
        pfpre->next=p;
        if(phead == NULL)
            phead = p;
        pfpre = p;
    }
    return phead;
}
void main() {
    LIST *p;
    p = buildlist(5);
    while(p->next != NULL)
```

```
p = p->next;
while(p!= NULL) {
printf("%d ",p->data);
p = p->prev;
}
}
```

三、问答题（共 10 分）

1、简述全局变量、静态全局变量、静态局部变量的异同。（6 分）

2、假定解决一个问题有两种算法程序，采用算法 A 的程序 A 和采用算法 B 的程序 B，它们在某一计算机上编译运行，已知问题的规模 n 和解决问题两个程序的运行时间 T(A)和 T(B)见下表:(us 表示微秒，ms 表示毫秒)(4 分)

n	1,000	2,000	4,000	8,000
T(A)	100us	410us	1.62ms	6.52ms
T(B)	1.1ms	2.2ms	4.41ms	8.83ms

请估计算法 A 与算法 B 的时间复杂度。

四、程序填空题（每空 2 分，共 24 分）

1、函数 f()用来计算 Fibonacci 数列(1、1、2、3、5、8、...)的第 n 项，函数 g()用来把 Fibonacci 的前 n 项填入数组 a 中，函数 main()输出 Fibonacci 的前 n 项。其中 f()、g()均为递归函数。

```
int f(int n){
    if(n==1 || n==2) return 1;
        return _____(1)_____;
    }
void g(int a[], int n){
    if(n>=2){
        g(a,n-1);
        a[n-1]=f(n);
    }
    else
        _____(2)_____;
    }
void main(){
    int a[100],i,n;
    scanf("%d", &n);
    _____(3)_____;
    for(i=0; i<n; i++)
        printf("%d ", a[i]);
    }
```

2、一群猴子要选新猴王，新猴王的选择方法是:让 n 只候选猴子围成一圈，从某位置起顺序编号为 1~n 号。从第 1 号开始报数（从 1 到 3），凡报到 3 的猴子即退出圈子，接着又从紧邻的下一只猴子开始同样的报数。如此不断循环，最后剩下的一只猴子就选为猴王。在空缺处填上正确的内容。

```
#include <stdio.h>
#define M 10
int find_next( int start, int monkey[][M], int n){
    int i=(start+1)%n;
    while( monkey[1][i]==0 )
        _____(4)_____;
    return i;
}
int main(){
    static int monkey[2][M]; /* monkey[0] 存储编号，monkey[1] 表示是否在列 */
    int i, count, n, start=0;
    scanf("%d",&n);
    for(i=0; i<n; i++) {
        monkey[0][i] = i+1;
        monkey[1][i] = 1; /* all the monkeys are in the circle */
    }
    for( count=n; count>1; count--) {
```

```

i=_____(5)_____;
i= find_next(i,monkey,n);
_____(6)_____;
start= find_next(i,monkey,n);
}
printf("The king is monkey[%d].\n",monkey[0][start]);
return 0;
}

```

3、以下是完成快速排序的函数，试在空缺处填入适当的内容。。

```

void quicksort(int list[],int m,int n)
{
    int key, i, j, k;
    if(m<n) {
        k=(m+n)/2;
        swap(&list[m],&list[k]);
        key = list[m];
        i = m+1;
        j = n;
        while(i<=j){
            while((i<=n)&&(list[i]<=key)) i++;
            while(_____(7)_____) j--;
            if(_____(8)_____) swap(&list[i],&list[j]);
        }
        _____(9)_____;
        quicksort(list,m,j-1);
        quicksort(list,j+1,n);
    }
}

```

4、函数 main()中，定义了迷宫地图 map，其中'='表示道路，'#'表示墙；数组 mark 用来标记 map 中对应位置是否已走过，走过标为 1，没走过则为 0；begin 是走迷宫的起始位置 0 行 0 列，end 表示迷宫的终点 4 行 4 列。函数 dig 使用递归法走迷宫，其中参数 rows 为迷宫的行数，cols 是迷宫的列数；当走到终点时 dig()返回 1,走不通时返回 0。dig()的递归过程如下：①先判断 begin 能否走，例如该位置已经走过或者坐标越界就不能走，不能走就返回 0；②把 begin 设为已走过，地图对应位置标成'*'；③以 begin 为中心，按左上右下 4 个方向尝试对它周围的 4 个位置进行 dig()；④若 4 个方向均走不通，则放弃 begin 这个点，回到 begin 前面那个点，程序的输出结果如下：

```

* * # = #
# * # = #
# * * * #
# = # * *
= = # # *

```

```

typedef struct point{
    int row, col;

```

```

}POS;
int dig(char *map[5], char mark[5][5], int rows, int cols, POS begin, POS end) {
int k, d[4][2]={0,-1}, {-1,0}, {0,1}, {1,0}};
int r=begin.row, c=begin.col;
if(r < 0 || r >=rows || c < 0 || c >=cols) return 0;
if(_____(10)_____) return 0;
if(mark[r][c] == 1) return 0;
mark[r][c] = 1;
map[r][c] = '*';
if(r == end.row && c == end.col) return 1;
for(k=0;k<4;k++) {
    _____(11)_____;
begin.col = c+d[k][1];
if(dig(map, mark, rows, cols, begin, end)==1) return 1;
    }
    _____(12)_____;
    return 0;
}
void main() {
char *map[5]={
"==##",
"#-##",
"#===#",
"#-##=",
"==##="
};
static char mark[5][5];
POS begin={0,0}, end={4,4};
int i;
dig(map, mark, 5, 5, begin, end);
for(i=0;i<5;i++) {
puts(map[i]);
}
}

```


五、算法设计（共 18 分）

1、通过 malloc()函数动态申请二维数组所需的存储空间，采用如下方法来生成具有 m 行 n 列的动态大小的二维整型数组(m 与 n 为变量，其值要在运行时确定):

```
int **pa = (int**)malloc(m*sizeof(int*));    /*生成动态数组，该数组具有 m 个 int 类型的指针分量*/
```

```
for(i=0; i<m; i++)
```

```
pa[i] = (int*)malloc (n*sizeof(int));    /*pa[i]为第 i 行首地址，为每行分配了 n 个 int 型存储空间*/
```

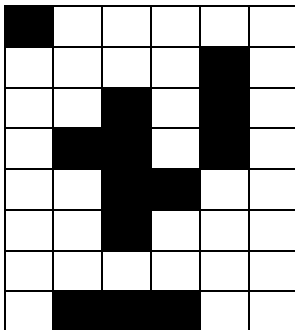
动态申请三个二维整型数组 A、B、C，其中 A 为 m 行 n 列，B 为 n 行 m 列，C 为 m 行 m 列，正整数 m 和 n 由用户从键盘输入。

(1)实现输入一个二维整型数组的函数，原型为: void input(int **pa, int row, int col); (4 分)

(2)实现矩阵 A 与 B 相乘的函数，原型为: void mmult(int **pc, int **pa, int row, int col, int **pb); (4 分)

(3)(2)中相乘算法的时间复杂度是多少? (2 分)

2、下面是一个由黑白格子构成的正方形网格图。在图中，如果两个网格都是黑色的，并且能够从一个网格连续移动到另一个网格，则称这两个网格属于一个块。(移动可以按照水平、竖直或对角线方向，移动过程中只能经过黑色格子。)



假设已有 C 语言程序:

```
#define BLACK 1/*黑色方格*/
```

```
#define WHITE 0/*白色方格*/
```

```
#define HEIGHT 8
```

```
#define WIDTH 6/*宽度为 6，高度为 8 的图*/
```

```
#include ../头文件已正确包含*/
```

```
void generateRandomMap( int map[HEIGHT][WIDTH]);
```

```
int map[HEIGHT][WIDTH];
```

```
void eraseBlob(int row; int col);
```

```
int main(void) {
```

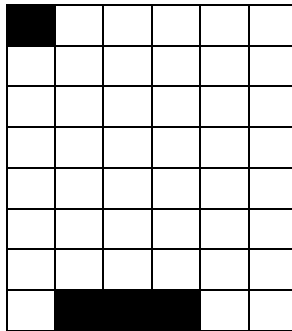
```

int row, int col;
generateRandomMap(map);
scanf("%d%d", &row, &col);
eraseBlob(row, col);
return 0;
}
void generateRandomMap( int map[HEIGHT][WIDTH])
{ /* ... */ }
void eraseBlob(int row, int col)
{ /* .. */ }

```

其中，二维的黑白网格图用数组 `map` 表示，当表示黑色网格时，元素值为 1；表示白色网格时，元素值为 0。`generateRandomMap()` 函数可以在 `map` 数组中生成一个随机的黑白网格图。

`eraseBlob()` 函数会将 `map[row][col]` 所在的块变为白色。例如，当 `map` 表示上图所示的网格图时，调用 `eraseBlob(2,2)` 后，`map` 变为下图：



(1) 简要描述 `eraseBlob()` 函数的实现思路；(4 分)

(2) 写出该函数的定义。(4 分)

2015-2016 学年第二学期期末考试 A 卷参考答案

一、单选题（每小题 3 分，共 30 分）

1、【正解】C

【解析】预处理 `#include` 是 ANSI 标准定义的 C 语言预处理指令，表示使编译程序将另一源文件嵌入到带有 `#include` 的源文件中。同时 `#include <>` 指的是先从标准 C 语言库中读取某头文件内容，如果未搜索到，则搜索个人文件夹的自定义头文件。因此 C 正确。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

2、【正解】A

【解析】函数调用时，要求相对应的形式参数和实际参数的数据类型必须一致。二维数组作参数调用时只有两种方式，用二维数组名作为实际参数或者用二维数组起始地址的一级地址形式作为实际参数。而在 A 选项中，语句 `aver(a,200)`，使用的数组名 `a` 表示的是二维数组的起始地址的二级地址，不符合调用规则。因此，选择 A 选项。

【考点延伸】《考试宝典》专题八 8.2 函数的调用、二维数组作函数的参数

3、【正解】D

【解析】函数的递归调用可以看成是一种特殊的函数嵌套调用。但递归调用不能无限制进行下去，即这种特殊的自己对自己的嵌套调用总会在某种条件下结束。因此，选择 C 选项。

【考点延伸】《考试宝典》专题八 8.2 函数的调用

4、【正解】C

【解析】考察指向函数的指针变量作函数的形式参数。指向函数的指针变量的定义形式为：[存储类别符] 数据类型名(*指针变量名)(形参表)。同一程序中，指针变量名不能重复。此题中，指向 `sort()` 函数的指针变量，应该定义为：`void(*q)(int a[],int size)`，因此选 C。

【考点延伸】《考试宝典》专题八 8.2 函数的调用、指向函数的指针变量

5、【正解】C

【解析】宏定义的一般形式为：`#define` 宏标识符(形参表) 表达式样式字符串。此题定义的 `MAX` 格式应为 `MAX(a,b)`，没有空格；表达式样式字符串为 `((a)>(b)?(a):(b))`，表示比较参数 `a` 和 `b` 的大小，返回较大的数。因此，选择 C 选项。

【考点延伸】《考试宝典》专题二 2.2 运算符和表达式、专题八 8.4 预处理命令

6、【正解】B

【解析】考察使用 `typedef` 构造数据类型别名。`typedef int *ptr;` 语句表示构造了一个指向整型整数的指针变量数据，名称为 `ptr`。`p2` 是 `ptr` 类型的数据，因此其实际上是一个指向整数的指针。选择 B 选项。

【考点延伸】《考试宝典》`typedef` 关键字的简单应用

7、【正解】D

【解析】考察结构体类型数组与指针的关系。指针变量 `p` 指向结构体某一数组元素（假设为 `s[2]`），表示它的二级成员分量 `Math` 有三种形式：`s[2].score.Math`、`(*p).score.Math`、`p->score.Math`。同理可写出其二级成员分量 `Total` 的三种表示方法，分别为：`s[2].score.Total`、`(*p).score.Total`、`p->score.Total`。因此，D 选项正确。

【考点延伸】《考试宝典》专题九 9.2 结构数组与结构指针

8、【正解】D

【解析】指针 `p` 指向链表中的一个结点(非首、尾结点)，如果要删除 `p` 结点，则首先要将 `p->next` 的内容复制下来，使中间变量指针 `n` 指向 `p->next`；然后将 `p->next=p->next->next;free(n);`，接着重新建立结点连接到之前的下一结点，实现删除当前结点的效果。因此，选择 D 选项。

【考点延伸】《考试宝典》专题九 9.7 用指针处理链表

9、【正解】B

【解析】选择排序在最坏情况下的时间复杂度为 $O(N^2)$ 。因此，B 选项正确。

【考点延伸】《考试宝典》选择排序的基本知识

10、【正解】A

【解析】归并非递归的排序算法是采用二叉堆排序的概念，自底向上，分层排序。第一趟归并后结果为：3 4 7 8 0 9 1 2 5 6；接着第二趟以间隔为 2，因此是[3 4 7 8]一组归并（结果顺序不变），[0 9 1 2]一组归并（结果为：0 1 2 9），[5 6]一组（结果顺序不变）。因此，第二趟归并结束后的结果为：3 4 7 8 0 1 2 9 5 6，选择 A 选项。

【考点延伸】《考试宝典》归并排序

二、改错题（共 18 分）

1、【正解】将 StudentListNode *next 改为：struct StudentListNode *next;

【解析】C 语言中需要使用关键字 struct 定义结构体类型，除非先用 typedef 为这种结构体数据类型取过别名，才能接着用别名定义相同类型的变量。

【考点延伸】《考试宝典》专题九 9.1 结构的定义与使用

2、【正解】将 printf("%d", num % base);change(num/base,base);的顺序交换，改为：change(num/base, base);printf("%d", num % base);

【解析】根据 if 语句，判断 num 大于 base 的值，如果是则进行 if 内语句，如果不是则直接输出该值。因此 if 内语句应起到更新的作用。调换顺序后才能实现从最高数位依次输出个数位上的值，直到末位。

【考点延伸】《考试宝典》专题八 8.2 函数的调用

3、【正解】删去#define DEBUG 0,
或将#ifdef DEBUG 改为#if DEBUG

【解析】#if 的一般含义是：如果#if 后面的常量表达式为 true，则编译它与#endif 之间的代码，否则跳过这些代码，#else 建立另一选择（在#if 失败的情况下）；而#ifdef 表示如果有定义，是条件编译的另一种方法。因此在这道题中，为了表达正确的逻辑需要删去#define DEBUG 0，或将#ifdef DEBUG 改为#if DEBUG。

【考点延伸】《考试宝典》专题八 8.4 预处理命令

4、【正解】将 return Find(x+m+1,n,v);改为：return Find(x+m+1,n-m-1,v);

【解析】题中采取的查找方法为：从数组 x 的中位数开始比较，如果该值比要查找的数值小，则更新比较的范围为中位数右边的数，即更新指针参数指向 x[m+1]，调整新的比较范围长度为 n-m-1，查找的值依然为 v；如果该值比要查找的数值小，则更新比较的范围为中位数左边的数。据此，应将 if(x[m]<v)后的语句 return Find(x+m+1,n,v);改为 return Find(x+m+1,n-m-1,v);

【考点延伸】《考试宝典》专题八 8.2 函数的调用

5、【正解】删去 free(a.name)

或将 free(a.name);printf("%s %d\n", b.name, b.score);的顺序交换，改为：printf("%s %d\n", b.name, b.score);free(a.name);

【解析】malloc()和 free()属于内存管理的两个函数，malloc 是申请内存的，free 是释放内存的。此题中 a.name 与 b.name 均是字符型指针变量，且指向的是同一地址（存储空间）的字符变量，如果先执行 free(a.name);，这一存储空间被释放，那么结果输出的 b.name 就会是乱码。因此，选择删去 free(a.name)，或将 free(a.name);printf("%s %d\n", b.name, b.score);的顺序交换，改为：printf("%s %d\n", b.name, b.score);free(a.name);

【考点延伸】《考试宝典》C 语言常见函数 malloc()和 free()

6、【正解】把 pfore->next = p;改为：
if(pfore != NULL)
pfore->next = p;

【解析】与单链表不同，双链表创建过程中，每创建一个新节点，都要与其前驱节点建立两次联系，分别是：将新节点的 prev 指针指向直接前驱节点；将直接前驱节点的 next 指针指向新节点。此题中在创建链表过程中有一个逻辑错误，需要判断当临时指针 pfore 不指向 NULL 时，再执行 pfore->next = p;进行结点的连接。

【考点延伸】《考试宝典》专题九 9.7 用指针处理链表

三、问答题（共 10 分）

1、【解析】相同点：生命周期都是全程的，(初始化缺省值为 0，变量只能被初始化一次)；
不同点：静态局部变量的作用域在函数内部，静态全局变量的作用域是模块内部自定义开始，全局变量的作用域还可以扩展到其它模块，只要它们采用 `extern` 声明。

【考点延伸】《考试宝典》专题七 7.2 变量的存储类型

2、【解析】从表上数据的“趋势”看：

$f_A(2n) \approx 4f_A(n)$ 由此可知，算法 A 的时间复杂度 $T(A) \approx O(n^2)$ ；

$f_B(2n) \approx 2f_B(n)$ 由此可知，算法 B 的时间复杂度 $T(B) \approx O(n)$ 。

而在计算时间复杂度时，先找出 $T(n)$ 的同数量级 ($1, \log_2 n, n, n \log_2 n, n^2, n^3, 2n, n!$)，一般 $f(n)$ = 该数量级。若 $T(n)/f(n)$ 求极限可得到一常数 c ，时间复杂度 $T(n) = O(f(n))$ ，反之利用 $f(n)$ 可估算 $T(n)$ 。

【考点延伸】《考试宝典》C 语言的时间复杂度

四、填空题（每小题 2 分，共 24 分）

1、【正解】(1) $f(n-2)+f(n-1)$

(2) $a[0]=f[n]$

(3) $g(a,n)$

【解析】函数 $f()$ 用来求第 n 项的值，因此 $f()$ 函数中，当 n 为 1 和 2，值均返回 1；当不为第 1 或 2 项时，第 n 项的值为第 $n-2$ 和 $n-1$ 项值之和，因此返回 $f(n-2)+f(n-1)$ 。函数 $g()$ 用来把 Fibonacci 的前 n 项填入数组 a 中，根据 `if` 内的语句逻辑判断，`else` 后应该是为数组 a 初项赋值，因此第二空为 $a[0]=f[n]$ 。接下来需要在主函数中调用 $g()$ 函数，参数一一对应，正确语句为： $g(a,n)$ 。

【考点延伸】《考试宝典》专题八 8.2 函数的调用

2、【正解】(4) $i=(i+1)\%n$

(5) `find_next(start,monkey,n)`

(6) `monkey[1][i] = 0`

【解析】函数 `find_next()` 用来排除淘汰了的猴子，找到下一个报数的猴子编号，因此第四空应为 $i=(i+1)\%n$ (递增更新猴子的编号，为了接下来判断其是否淘汰，直到找出下一个还可报数的猴子编号 i)。第五空应在主函数中调用 `find_next()` 函数，找到第二个报数编号。 $i=\text{find_next}(i,\text{monkey},n)$ ；语句表示找到第三个报数的猴子编号，接下来第六空，应淘汰次编号的猴子，为 `monkey[1][i] = 0`。接下来又接着重复报数过程，直到最后一个。

【考点延伸】《考试宝典》专题八 8.2 函数的调用

3、【正解】(7) $(j \geq m) \&\& (\text{list}[j] > \text{key})$

(8) $i < j$ 。

(9) `swap(&list[m],&list[j])`

【解析】快速排序每次把数组分成两部分和中间的一个划分值，通过两部分内的值分别和划分值 (key) 比较大小，再进行顺序的调整，实现快速排序的效果。第七空所在行的语句应该预期实现从右至中间依次比较该数值与划分数的大小，直到找出比划分数小的值。第八空表示交换前两行找出的两个数值，实现顺序的部分调整，这里缺少判断条件应为： $i < j$ 。第九空为调整划分值。

【考点延伸】《考试宝典》专题四 4.1 `while` 函数、专题八 8.2 函数的调用、快速排序函数

4、【正解】(10) `map[r][c] == '#'`

(11) `begin.row = r+d[k][0]`

(12) `map[r][c] = ' '`

【解析】第十空表示遇到墙走不通，返回 0，因此判断的条件为 `map[r][c] == '#'`。第十一空表示调整新的行坐标，结合下一行语句和数组 d 来看，这里应为 `begin.row = r+d[k][0]`，表示依次从此点的左、上、右、下点新出发。第十二空需要对已走过的但不是最终路径的路进行还原。

【考点延伸】《考试宝典》专题八 8.2 函数的调用

五、算法设计（共 18 分）

1、【解析】(1) `void input(int **pa, int row, int col)` //按题目要求，定义一个二维整型数组的函数

```

{
int i, j, k;
for(i=0; i<row; i++)      //变换行数
for(j=0; j<col; j++)      //变换列数
scanf("%d",pa[i+j]); //标准化输入该数值
}
//对于每一个一维数组 pa[i]或*(pa+i), pa[i+j] 或 *(pa+i+j) 表示该一维数组 pa[i]中第 j 个元素的地址,
即&pa[i][j];

```

(2) void mmult(int **pc, int **pa, int row, int col, int **pb) //定义函数

```

{
    int i, j, k;
    for(i=0; i<row; ++i){ //变换行
        for(j=0; j<col; ++j){ //变换列
pc[i][j]= 0; //新二维数组初始化
for(k=0; k< col; ++k){
pc[i][j] +=pa[i][k]*pb[k][j]; //矩阵 A 与 B 相乘, 生成新的矩阵
}
}
}
}
}

```

(3) 该算法的时间复杂度为 $O(M*N*N)$

根据 for 循环, row 值为 M, col 值为 N。这里嵌套了三个 for 循环因此时间复杂度为 $O(M*N*N)$ 。

【考点延伸】《考试宝典》专题六 6.5 指针数组与数组指针、C 语言程序的时间复杂度

2、【解析】(1) 调用 eraseBlob 函数后, 先判断初始点的方块颜色, 如果为黑色, 则改其颜色为白色, 接着再从初始点开始依次遍历其上、下、左、右、左上、左下、右上、右下的方块。如果遇到这个方块的颜色为黑色, 则变其为白, 再从当前方块开始按照其上、下、左、右、左上、左下、右上、右下的顺序遍历它周围的方块。(先一个方向走到头, 再返回上一层, 类似深度搜索) 反复重复这一过程, 直到所有能遍历得到得方块结束。

(2) void eraseBlob(int row, int col){

```

    if(row >= 0 && <HEIGHT && col>=0 && col<WIDTH){ //判断是否越界
        if(map[row][col] == BLACK){ //判断是否为黑色方块
            map[row][col] = WHITE; //将黑色方块变为白色, 接着遍历其周围
            eraseBlob(row-1,col); //上
            eraseBlob(row+1,col); //下
            eraseBlob(row,col-1); //左
            eraseBlob(row,col+1); //右
            eraseBlob(row-1,col-1); //左上
            eraseBlob(row+1,col-1); //左下
            eraseBlob(row-1,col+1); //右上
            eraseBlob(row+1,col+1); //右下
        }
    }
}

```

【考点延伸】《考试宝典》专题八 函数