

快快编程模拟试卷 1 答案解析

一、单项选择题（共 15 题，每题 2 分，共计 30 分，每题有且仅有一个正确选项）

1. 如果 128 种颜色用二进制编码来表示，至少需要（ ）位。
A. 5 B. 6 C. 7 D. 8

答案：C

解析：每一位可以表示 0 或 1，从 0000000 到 1111111，7 位一共可以表示 $2^7=128$ 种。

2. 若 $12 \times 25 = 311$ 成立，则用的是（ ）进制。
A. 11 B. 8 C. 7 D. 9

答案：D

解析： $(1 \times n + 2) \times (2 \times n + 5) = 3n^2 + 1 \times n + 1$ 。 $n^2 - 8n - 9 = 0$ ，解得 $n=9$ 或 $n=-1$ 。

3. 一个袋子里装了 100 个苹果，100 个香蕉，100 个橘子，100 个梨子。从袋子中取出一个水果需要 1 分钟，那么需要分钟就能肯定至少已经拿出 1 打（12 个）相同种类的水果（ ）种。
A. 12 B. 13 C. 45 D. 101

答案：C

解析：根据鸽巢原理， $4 \times 11 + 1 = 45$ 。

4. 下列程序段的时间复杂度是（ ）。

```
cnt=0;
for(int i=1; i<n; i*=2)
    for(int j=1; j<n; j++)
        cnt++;
```

- A. $O(\log_2(n))$ B. $O(n)$ C. $O(n \times \log_2(n))$ D. $O(n^2)$

答案：C

解释：时间复杂度计算。对外层的 for 循环，设其执行次数为 t ，则 $2^t < n$ ，即 $t < \log_2(n)$ 。内层 for 循环需要执行 n 次，于是整段程序的时间复杂度是 $O(n \times \log_2(n))$ 。

5. 以下是 32 位机器与 64 位机器区别的是（ ）。
- A. 硬盘大小不同
B. 显示器分辨率不同
C. 操作系统版本号不同

D. 寻址空间不同

答案：D

解析：32 位机器与 64 位机器的区别是寻址空间不同。

6. 如果一棵二叉树的先序遍历是 ACDBEFG，中序遍历是 DCAEBFG，那么它的后序遍历是 ()。

- A. DCEGFBA
- B. DCAEFGB
- C. DCEFGBA
- D. DCAEGFB

答案：A

解析：先序：根-左-右。中序：左-根-右。先序中首个结点即为根（在本题中为 A），然后通过中序里根 A 的所在位置，划分左右子树（根分左右）；再根据左右子树的结点个数，去划分得到先序的左右子树。递归进行上述步骤。可画出树的形态，最后再求后序遍历。

7. 如果开始时计算机处于小写输入状态，现在有一只小狗反复按照 CapsLock、字母键 A、字母键 B、字母键 C、字母键 D 的顺序来回按键，即 CapsLock、A、B、C、D、CapsLock、A、B、C、D、CapsLock、A、B、C、D、……，屏幕上输出的第 2020 个字符是字母 ()。

- A. 大写 A
- B. 大写 B
- C. 大写 C
- D. 大写 D

答案：D

解析：CapsLock 是大小写切换。所以，每组一共八个字母，即 ABCDabcd。2020%8=4，第 2020 个输出的是大写 D。

8. 如果进栈序列位 e1,e2,e3,e4，则不可能的出栈序列是：()。

- A. e2, e4, e3, e1
- B. e4, e3, e2, e1
- C. e1, e2, e3, e4
- D. e3, e1, e4, e2

答案：D

解析：e3 先出栈，则 e1 不可能在 e2 之前出栈，D 错误。

9. 有如下程序:

```
#include <iostream>
using namespace std;
int a[3][3];
int main(){
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            a[i][j]=i*3+j+1;
    for(int i=1;i<3;i++){
        for(int j=0;j<2;j++)
            cout<< a[j][i];
        cout << endl;
    }
```

```

    }
    return 0;
}

```

运行后的输出结果是 ()。

- A. 2536 B. 2356 C. 4758 D. 4578

答案：A

解析：注意输出的是 $a[j][i]$ ，而不是 $a[i][j]$ 。所以依次输出的是 $a[0][1]$ 、 $a[1][1]$ 、 $a[0][2]$ 、 $a[1][2]$ 。

10. 8 名男生和 4 名女生围绕圆桌就坐,任意两个女生不相邻的坐法有 () 种。

- A. 67737600 B. 8467200 C. 40320 D. 352800

答案：B

解析：男生圆排列，按圆排列的公式可得 $8!/8=7!$ 。然后固定一个男生，把女生插空到 8 名男生之间： $A(8,4)$ 。最后，方案数为： $7!*A(8,4)=8467200$ 。

11. 关于下面说法，错误的一项是 ()。

- A. ASCII 码是一种字符编码,常用 7 位码
 B. 程序设计的三种基本结构是顺序、选择、循环
 C. 刷新率、CPI、DPI 均可以用于描述鼠标性能
 D. 目前世界上最大的计算机互联网络是 IBM 网

答案：D

解析：Internet 是目前世界上覆盖范围最广、使用者最多、最成功的计算机互联网络。

12. 小红课桌上有 5 本书，每本书都放有一个写着书名的书签。调皮的小王把小红的书签全部打乱了。那么，每个书签都没有插到对应的书中的可能性总共有 () 种。

- A. 38 B. 40 C. 44 D. 48

答案：C

解析：经典的错排问题，假设 n 本书和书签全打乱的可能性一共是 $f(n)$ 种，那么遵循如下递推公式 $f(n) = (n-1) * (f(n-1)+f(n-2))$ ，所以 $f(5)=44$ 。

13. 设 G 是有 7 个结点的完全图，要得到一棵生成树，需要从 G 中删去 () 条边。

- A. 10 B. 12 C. 15 D. 18

答案：C

解析：7 个点的完全图有 $7*6/2=21$ 条边，而 7 个点的树有 $7-1=6$ 条边，因此需要删去 $21-6=15$ 条边。

14. $46!$ 的计算结果，尾数总共有 () 个零。

- A. 9 B. 10 C. 11 D. 12

答案：B

解析：求 46! 尾数总共有几个零，也就是求 46! 中 2*5 的因子的个数。由于因子 2 的数量远多于 5 的个数，所以就转换成求 5 的因子个数，1 到 46 中有 5、10、15、20、25 (5*5)、30、35、40、45，总计 10 个。

15. 定义语句“double * array[8]”的含义正确的是 ()。

- A. array 是一个指针，它指向一个数组，数组的元素是双精度浮点型。
- B. array 是一个数组，数组的每一个元素是指向双精度浮点型数据的指针
- C. C++ 语言中不允许这样的定义语句
- D. 以上都不对

答案：B

解析：根据数组定义：array 是一个数组，包含了 8 个 double 型指针。

二、阅读程序（程序输入不超过数组或字符串定义的范围：判断题正确填√，错

误填×；除特殊说明外，判断题 1.5 分，选择题 3 分，共计 40 分）

1.

1	#include <iostream>
2	using namespace std;
3	int main() {
4	int max, min, sum, cnt = 0;
5	int temp;
6	cin >> temp;
7	if (temp == 0) return 0;
8	max = min = sum = temp;
9	cnt++;
10	while (temp != 0) {
11	cin >> temp;
12	if (temp != 0) {
13	sum += temp;
14	cnt++;
15	if (temp > max) max = temp;
16	if (temp < min) min = temp;
17	}
18	}
19	cout << cnt << " " << min << " " << max << " " << sum/cnt << endl;
20	return 0;
21	}

判断题：

1. 本程序统计了输入数字个数、最大值、最小值、平均值，输入遇-1 结束。()

答案：F

解析：遇 0 结束

2. 对任意的整数值 x ，只需要将第 7 行中 if 条件判断修改为 $\text{temp}==x$ ，就可以将本程序功能变成输入遇值 x 结束。()

答案：F

解析：如果只修改第 7 行，那么只要整个输入序列的首个数字不是 x ，程序就不会在第 7 行结束。

3. 去掉第 8 行的连续赋值语句，或者忘记对第 4 行的整型变量 cnt 初始化为 0，这两种情况都可能影响程序最终输出结果。()

答案：T

解析：局部变量需要初始化。

4. 将 15 行 if 条件判断语句改成 $\text{temp} \geq \text{max}$ ，并且把 16 行 if 条件判断语句改成 $\text{temp} \leq \text{min}$ ，对程序结果没有任何影响。()

答案：T

解析：打擂台过程中，取等号对选最大最小值没有影响。

选择题：

5. 若输入 0 1 2 3 4 5，程序运行到第 () 行结束。

A. 6 B. 7 C. 10 D. 20

答案：B

解析：输出的首个数字即为 0，直接在第 7 行结束。

6. 若输入 1 2 3 4 6 0 8 9，程序最终输出结果是 ()

A. 5 1 6 3.2 B. 8 1 9 4 C. 6 1 6 3 D. 5 1 6 3

答案：D

解析：当输出 0 时，while 循环结束。 cnt 为 5， min 为 1， max 为 6，平均值是 $16/5=3$ 。

2.

1	#include <stdio.h>
2	int gcd(int a, int b){
3	if(b == 0) return a;
4	return gcd(b,a%b);
5	}
6	void work(int a, int b) {
7	int s[10009], t[10009], i = 0, d = 1, j;
8	while(1) {
9	if(a == 0) break;

10	a *= 10; //除法借位
11	t[i] = a;
12	s[i] = a / b;
13	a %= b;
14	for(j = 0; j < i; ++j)
15	if(t[j] == t[i]) {
16	d = 0;
17	break;
18	}
19	if(d == 0) break;
20	printf("%d", s[i]);
21	++i;
22	}
23	}
24	int main() {
25	int a, b, g;
26	scanf("%d%d", &a, &b);
27	if(a > b) g = gcd(a, b);
28	else g = gcd(b, a);
29	a /= g;
30	b /= g;
31	printf("%d.", a/b); //格式化输出一个整数和一个小数点
32	a %= b;
33	work(a, b);
34	return 0;
35	}

本程序输入一个分数的正整数分子 a 和正整数分母 b，将其转换为小数形式输出。

判断题：

1. 函数 gcd 使用了递归的方法，返回参数 a 和 b 的最小公倍数。()

答案：F

解析：返回的是 a 和 b 的最大公约数。

2. 33 行作为 work 函数的参数传入 a 和 b，一定满足 a<b 且 a≠0()

答案：F

解析：经过 a%=b 的计算，a 完全可能等于 0。

3. 去掉 14-19 行的代码，将增加程序发生运行时错误的风险。()

答案：F

解析：如果遇到无限循环小数，while 循环没有设计退出机制，数组很快会越界。但要注意，如果循环节过大，本身也可能有数组越界风险。

4. 去掉 27-30 行，程序输出结果一定发生改变。()

答案：F

解析：程序输出结果不变，把分子和分母化约到互质形式，对程序结果无影响。

选择题：

5. 若输入 91 13，输出结果是()。

A. 7 B. 7. C. 7.0 D. 7.00

答案：B

解析：91/13=7，根据 printf 函数，首先输出 7.。接下来 a 被化约到 0，因此调用函数是 work(0, 13)。进入函数，第 9 行直接 break，没有输出。

6. 以下哪个结果是程序可能输出的 ()。

A. 10.00 B. 1.571428571428 C. 1.5714285 D. 1.571428

答案：D

解析：本程序输出到第 1 个循环节结束为止。A 选项中 00 不可能构成一个循环节。而 B、C 两个选项都超过了一个循环节。

3.

```
1  #include <iostream>
2  using namespace std;
3  int solve(int n, int m){
4      int i, sum;
5      if (m == 1) return 1;
6      sum = 0;
7      for (i = 1; i < n; i++)
8          sum += solve(i, m - 1);
9      return sum;
10 }
11 int main(){
12     int n, m;
13     cin>>n>>m;
14     cout<<solve(n, m)<<endl;
15     return 0;
16 }
```

判断题：

1. 程序第三行改写成 solve(int m, int n)，程序运行结果不变。()

答案：F

解析：将两个参数互换位置，程序运行结果受影响。

2. 去掉程序第 6 行后，程序运行结果不变。()

答案：F

解析：sum 是函数内部局部变量，不初始化则是垃圾值。

选择题：

3. 若输入的 $n < m$ ，则输出结果为 ()。

A. 0 B. n C. m D. $m + (m + 1) + \dots + (n - 1)$

答案：A

解析：递归调用 solve(n,m)，每深入一层 m 减 1，n 至少减 1 (for 从 1 到 n-1)，最终 n 减少到 1 无法进入 for 时，m 必然还大于 1，没有返回 1，因此当 $n < m$ 时必然返回 sum=0。

4. 输入 4 3，程序第 8 行共循环 () 次。

A. 3 B. 6 C. 9 D. 12

答案：B

解析：第一层：分别调用 solve(1,2), solve(2,2), solve(3,2)，一共是 3 次。第二层：solve(1,2) 循环 0 次；solve(2,2) 循环 1 次 (只有 solve(1,1))，solve(3,2) 循环 2 次 (对应 solve(1,1) 和 solve(2,1))。

5. 输入 7 4，输出是 ()

A. 35 B. 10 C. 20 D. 15

答案：C

解析：需要结合讲义中方法：递归转换为二维表格递推。

6. (4 分) 以下四组输出数据中，输出最大的是 ()。

A. 6 3 B. 5 3 C. 7 5 D. 8 7

答案：C

解析：需要结合讲义中方法：递归转换为二维表格递推。

三、完善程序 (单选题，每小题 3 分，共计 30 分)

1. (活动选择) 学校在最近几天有 n 个活动，这些活动都需要使用学校的大礼堂，在同一时间，礼堂只能被一个活动使用。由于有些活动时间上有冲突，学校办公人员只好让一些活动放弃使用礼堂。现给出 n 个活动使用礼堂的起始时间 $begin_i$ 和结束时间 end_i ($begin_i < end_i$)。现程序在 第一行输入一个整数 n ($n \leq 1000$)，接下来 n 行，每行两个整数，第 1 个是 $begin_i$ ，第 2 个是 end_i ($begin_i < end_i \leq 32767$)，程序输出最多能安排的活动个数。

1	#include<iostream>
2	using namespace std;
3	int n, begin[1009], end[1009];
4	void init() {

5	cin >> n;
6	for (int i = 1; i <= n; i++)
7	cin >> begin[i] >> end[i];
8	}
9	void qsort(int x, int y) {
10	int i, j, mid, t;
11	i = ____ (1) ____;
12	j = y;
13	mid = ____ (2) ____;
14	while(i <= j) {
15	while(end[i] < mid) i++;
16	while(____ (3) ____) j--;
17	if(i <= j) {
18	t = end[j];
19	end[j] = end[i];
20	end[i] = t;
21	t = begin[j];
22	____ (4) ____;
23	begin[i] = t;
24	i++;
25	j--;
26	}
27	}
28	if(x < j) qsort(x, j);
29	if(i < y) qsort(i, y);
30	}
31	void solve() {
32	int ans = 0;
33	for(int i = 1, t = -1; i <= n; i++)
34	if(begin[i] >= ____ (5) ____) {
35	ans++;
36	t = end[i];
37	}
38	cout << ans << endl;
39	}
40	int main() {
41	init();
42	qsort(1, n);
43	solve();
44	return 0;
45	}

1. (1) 处应填 ()。

A. 0 B. x C. mid D. x

答案：B/D

解析：将游标 i 设置为本轮的左边界 x 。

2. (2) 处应填 ()。

A. $\text{end}[(x+y)/2]$ B. $\text{begin}[(x+y)/2]$ C. $(x+y)/2$ D. $\text{begin}[(x+y)/2] + \text{end}[(x+y)/2]$

答案：A

解析：选最中间的活动结束值 $\text{end}[(x+y)/2]$ 作为基准值 (pivot)。

3. (3) 处应填 ()。

A. $\text{end}[j] < \text{mid}$ B. $\text{end}[j] > \text{mid}$ C. $\text{begin}[j] < \text{mid}$ D. $\text{begin}[j] > \text{mid}$

答案：B

解析：快速排序 (quick sort) 基本操作。

4. (4) 处应填 ()。

A. $\text{begin}[j] = t$ B. $t = 0$ C. $\text{begin}[i] = \text{begin}[j]$ D. $\text{begin}[j] = \text{begin}[i]$

答案：D

解析：交换起始时间 $\text{begin}[i]$ 和 $\text{begin}[j]$ 的中间步骤。

5. (5) 处应填 ()。

A. ans B. $\text{end}[i]$ C. $n+1$ D. t

答案：D

解析：贪心策略。对排序后顺序遍历每个活动 i ，只要起始时间 \geq 上轮结束时间，则安排它，并更新最新结束时间 t 。

2. (细胞问题) 一个矩形阵列，由数字 0 到 9 组成。数字 1 到 9 是代表细胞。细胞的定义是，沿细胞数字上下左右四个方向，只要还是细胞数字，则被视为同一个细胞，求给出的矩形阵列的细胞个数。

如：阵列

4 10

0234500067

1034560500

2045600671

0000000089

有四个细胞。

1	<code>#include<iostream></code>
2	<code>#include<string></code>
3	<code>using namespace std;</code>
4	<code>int direction_x[4] = _____;</code>
5	<code>int direction_y[4] = {0, 1, 0, -1};</code>

```

6   int flag[100][100];
7   int num_of_cells = 0
8   int m,n;
9   void find(int start_x, int start_y) {
10      int x, y, t, w, i;
11      int queue[1000][2];
12      num_of_cells++;
13      flag[start_x][start_y] = 0;
14      t = 0;
15      w = 1;
16      queue[1][1] = start_x;
17      queue[1][2] = start_y;
18      do {
19          t++;
20          for (i = 0; i <= 3; i++) {
21              x = queue[t][1] + direction_x[i];
22              y = ____ (2) ____;
23              if((x >= 0)&&(x < m)&&(y >= 0)&&(y <=m)&&(flag[x][y])) {
24                  w++;
25                  queue[w][1] = x;
26                  queue[w][2] = y;
27                  flag[x][y] = 0;
28              }
29          }
30      }while(____ (3) ____);
31  }
32  int main() {
33      int i, j;
34      string s;
35      cin >> m >> n;
36      for(i = 0; i <= m - 1; i++)
37          for(j = 0; j <= n - 1; j++)
38              flag[i][j] = 1;
39      for(i = 0; i <= m - 1; i++) {
40          cin >> s;
41          for(j = 0; j <= n - 1; j++)
42              if(s[j] == '0')
43                  ____ (4) ____;
44      }
45      for(i = 0; i <= m - 1; i++)
46          for(j = 0; j <= n - 1; j++)
47              if(flag[i][j])
48                  ____ (5) ____;
49  }

```

50	cout<<"NUMBER of cells is "<< num_of_cells << endl;
51	return 0;
52	}

1. (1) 处应填 ()。

- A. {0,1,0,-1} B. {0,-1,0,1} C. {1,-1,-1,1} D. {-1,0,1,0}

答案：D

解析：d_x 和 d_y 联合起来作为连通四方向的移动数组。

2. (2) 处应填 ()。

- A. queue[t][2] + direction_y[i] B. queue[t][1] + direction_y[i]
C. queue[t][2] + direction_x[i] D. queue[1][t] + direction_y[i];

答案：A

解析：BFS 发洪水算法基本操作：移动行的同时移动列。

3. (3) 处应填 ()。

- A. w!=0 B. t>w C. t!=0 D. t<w

答案：D

解析：只要队列中还有元素 (t<w)，整个 BFS 搜索通过 while 持续进行，遍历完整个连通块并取消它们标记。

4. (4) 处应填 ()。

- A. flag[j]=0 B. flag[i][j]=0
C. flag[i][j]='0' D. flag[i][j]=1

答案：B

解析：如果这个点是字符 0，对应 flag 标记为 0。

5. (5) 处应填 ()。

- A. num_of_cells++ B. flag[i][j]=0
C. find(queue[i][1],queue[j][2]) D. find(i,j)

答案：D

解析：如果这个点是细胞点，就开始调用 find 函数进行遍历访问与之相连的整个连通块。