

## 提高组模拟题第三套试题及答案

1. 假设有以下定义: `int a[5]={1,2,3,4,5}`, `i=3`, `*p=a`, `*q=a`; 则不能正确执行的语句是 ( B )。

- A. `i=*p*q;`            B. `a=i;`  
C. `*p=(a+i);`        D. `i=*p**(q+2);`

解析: `a` 是数组不能这样直接赋值。

2. 下列不属于 CPU 的有 ( C )。

- A. 海思麒麟 990      B. Intel 酷睿 i7      C. 影驰 RTX2070      D. AMD Ryzen 7

解析:

C. 影驰 RTX2070 是显卡。

cpu: 麒麟是华为, intel AMD 这几个都是 CPU

3.  $(2019)_{10} + (2020)_8$  的结果是 ( B )。

- A.  $(3049)_{10}$             B.  $(BF3)_{16}$   
C.  $(101111110001)_2$     D.  $(5765)_8$

解析: 2020 八进制转十进制  $2*8^3 + 2*8^1 = 1040 + 2019 = 3059$

BF3 转十进制  $11*16^2 + 15*16^1 + 3*16^0 = 3059$ , 所以选 B。

4. 某二叉树的先序遍历序列和后序遍历序列正好相反, 则该二叉树具有的特征是 ( A )。

- A. 高度等于其结点数    B. 任一结点无左孩子  
C. 任一结点无右孩子    D. 空或只有一个结点

解析: B, C 不正确, D 空或者只有一个结点也是错的。肯定是每个结点只有一棵子树, 所以 A 是正确的。



5. 若有定义 `char x[] = "12345"`; `char y[] = { '1', '2', '3', '4', '5' }`; 则 ( B )。

- A. x 数组与 y 数组的所占内存空间相同      B. x 数组比 y 数组占的内存空间大  
C. x 数组比 y 数组所占内存空间小      D. x 数组等价于 y 数组

解析：字符串后有一个结束标记“\0”也占一个字节，故 x 是 6 个字节，y 是 5 个字节。

6. 公共汽车起点站于每小时的 10 分，30 分，55 分发车，该顾客不知发车时间，在每小时内的任一时刻随机到达车站，如果乘客到车站的时刻为发车时间，乘客不能坐上此时发车的公共汽车，则乘客候车时间的数学期望（准确到秒）是 ( D )。

- A. 8 分 40 秒      B. 15 分 20 秒      C. 22 分 30 秒      D. 10 分 25 秒

解析：设每小时节点 55 (min) ~ 10 (min) 的平均等候时间为  $\bar{t}_1$ ，10 (min) ~ 30 (min) 的平均等候时间为  $\bar{t}_2$ ，30 (min) ~ 55 (min) 的平均等候时间为  $\bar{t}_3$ ，乘客等候时间的数学期望是  $\bar{t}$ ，

$\bar{t}_1 = \frac{15}{2}$  (min)， $\bar{t}_2 = \frac{20}{2}$  (min)， $\bar{t}_3 = \frac{25}{2}$  (min)  $\bar{t} = \frac{15}{2} \times \frac{15}{60} + 10 \times \frac{20}{60} + \frac{25}{2} \times \frac{25}{60} = \frac{125}{12}$ ，即为 10 分 25 秒。

举 55 分到 10 分的那一段简单说明一下：这一段一共 15 分钟，在一个小时中所占的时间比重为 15/60，而这一段的平均时间显然就是中间值为 15/2，所以这一块的期望贡献为 (15/60)\*(15/2)。

7. 设要将序列 <Q, H, C, Y, P, A, M, S, R, D, F, X> 中的关键码按字母的升序重新排列，则 ( A ) 是以第一个元素为分界元素的快速排序一趟扫描的结果。

- A. F, H, C, D, P, A, M, Q, R, S, Y, X      B. P, A, C, S, Q, D, F, X, R, H, M, Y  
C. A, D, C, R, F, Q, M, S, Y, P, H, X      D. H, C, Q, P, A, M, S, R, D, F, X, Y

解析：

该题可以直接针对选项去排除，第一个元素是 Q，看选项中 Q 之前的元素是否均小于 Q，之后的元

素是否都大于 Q。如果仅有一个选项满足，则可以选择，如果有多个满足这样条件的选项，那么再按快排的方法，看看其他元素位置是否正确。

8. 设 G 是有 n 个结点、m 条边 ( $n \leq m$ ) 的连通图，必须删去 G 的 ( A ) 条边，才能使得 G 变成一棵树。

A.  $m-n+1$       B.  $m-n$       C.  $m+n+1$       D.  $n-m+1$

解析：n 个点要组成一棵树，应该是  $n-1$  条边，现有 m 条边，故需要删除  $m-n+1$  条边。

9. 将 2 个红球，1 个蓝球，1 个白球放到 10 个编号不同的盒子中去，每个盒子最多放一个球，有多少种放法 ( B )。

A. 5040      B. 2520      C. 1260      D. 420

解析：

先放蓝球和白球，有  $P_{10}^2$  种方法，再放红球有  $C_8^2$  种方法，故共有  $P_{10}^2 C_8^2$  种放法。

10. 一个家具公司生产桌子和椅子。现在有 113 个单位的木材。每张桌子要使用 20 个单位的木材，售价是 30 元；每张椅子要使用 16 个单位的木材，售价是 20 元。使用已有的木材生产桌椅（不一定要把木材用光），最多可以卖 ( C ) 元钱。

A. 140      B. 150      C. 160      D. 170

解析：

由于单位木材桌子价格更高，所以优先考虑生产桌子，生产 5 张桌子的话，剩余木材不够生产任何成品，可以卖 150 元，这个时候要合理利用剩余木材，可以少生产 1 张桌子，从而使得木材可以生产 2 张椅子，故最终可以卖出 160 元。

11. 给出 4 种排序：插入排序、冒泡排序、选择排序、快速排序。这 4 种排序的时间代价分别是 ( A )。

A.  $O(n^2)$ 、 $O(n^2)$ 、 $O(n^2)$ 、 $O(n \log n)$   
B.  $O(n^2)$ 、 $O(n^2)$ 、 $O(n^2)$ 、 $O(\log n)$   
C.  $O(n \log n)$ 、 $O(n^2)$ 、 $O(n^2)$ 、 $O(n \log n)$   
D.  $O(n \log n)$ 、 $O(n^2)$ 、 $O(n \log n)$ 、 $O(n \log n)$

解析：

插入排序、冒泡排序、选择排序复杂程度均为  $O(n^2)$ ，快速排序复杂程度为  $O(n \log n)$ ，桶排序复杂程度为  $O(n)$ 。

12. 以下数据结构中，哪一个不是线性结构？ ( B )

A. 广义表      B. 二叉树      C. 队列      D. 栈

解析：二叉树是树形结构，不属于线性结构。

13. 以下最短路算法中不能处理带有负权值的算法的是 ( A )。

- A. Dijkstra 算法                      B. Floyd 算法  
C. Bellman-Ford 算法                D. SPFA 算法

解析:

Dijkstra 算法只适用无负权值的单源最短路。Floyd 算法可以处理多源点, 适用有负权值, 但复杂度高。Bellman-ford 算法是求含负权图的单源最短路径的一种算法。SPFA 可以处理负边权, 但不能处理负权回路。

14. 栈 S 最多能容纳 4 个元素。现有 6 个元素按 1, 2, 3, 4, 5, 6 的顺序进栈, 问下列哪一个序列是可能的出栈序列? ( B )

- A. 5, 4, 3, 2, 1, 6                  B. 3, 2, 5, 4, 1, 6  
C. 2, 3, 5, 6, 1, 4                  D. 1, 4, 6, 5, 2, 3

解析:

A 肯定不可能, 如果 5 先出栈 1, 2, 3, 4, 5 都得入栈空间不够。D 的出栈顺序不对 2, 3 在栈里只能 3 比 2 先出栈, C 出栈序列也不对 1, 4 在栈里 1 不可能比 4 先出栈。

15. 平面上有三条平行直线, 每条直线上分别有 7, 5, 6 个点, 且不同直线上三个点都不在同一条直线上。问用这些点为顶点, 能组成 ( C ) 个不同四边形。

- A. 18            B. 210            C. 2250            D. 4500

解析:

假设 3 条平行线 a, b, c 上分别有 7, 5, 6 个点, 其中 a 上 7 个点中任取 2 点有  $(7*(7-1))/2=21$  种取法;

同理可知 b 上 5 个点中任取 2 点有 10 种取法; c 上 6 个点中任取 2 点有 15 种取法。

a 上任取 2 点, b 上任取 2 点, 可以组成的四边形个数是  $21*10=210$ ;

a 上任取 2 点, c 上任取 2 点, 可以组成的四边形个数是  $21*15=315$ ;

b 上任取 2 点, c 上任取 2 点, 可以组成的四边形个数是  $10*15=150$ ;

a 上任取 1 点, b 上任取 1 点, c 上取 2 点, 可以组成四边形个数是  $7*5*15=525$  (不同直线上三个点都不在同一条直线上, 所以肯定可以组成四边形);

a 上任取 1 点, b 上任取 2 点, c 上取 1 点, 可以组成四边形个数是  $7*10*6=420$ ;

a 上任取 2 点, b 上任取 1 点, c 上取 1 点, 可以组成四边形个数是  $21*5*6=630$ 。所以可以组成四边形个数是  $210+315+150+525+420+630=2250$ 。

阅读程序

```
1  #include<stdio>
2  using namespace std;
3  int findvall(int n)
4  {
5      int f;
6      if(n==0)return 1;
7      else
8      {
9          f=findvall(n/2);
10         return(n*f);
11     }
12 }
13 int main()
14 {
```

```

15     int n;
16     scanf("%d",&n);
17     printf("%d\n",findvall(n));
18     return 0;
19 }

```

(1) 第6行输出 if(n==0)改成 if(n==1)时,对于输入的正整数 n,输出结果不会改变。( )

答案 ✓

解析:  $f=findvall(n/2)$ ;递归结束边界改变,但这2个边界对于正整数 n 来说,值相同。

(2) 对于输入的正整数程序输出的值小于等于 n。( )

答案 ✗

解析: 输出的结果每进行一次递归,值会变大,很显然 n 越大,值远远超过 n。

(3) 如果输入的 n 是负数的话,该程序会出现死循环,所以该程序不能求解 n 是负数的情况。

( )

答案 ✗

解析: 递归每次调用,参数值除以 2,所以负数最终也能满足递归到 0 的结束条件。

(4) 如果多次运行该程序,并且输入的 n 是单调递增的正整数,那么每次输出的结果也是一个严格单调递增的数列。( )

答案 ✓

解析: 由于每次递归调用时均会乘以 n,所以随着 n 的增加,输出的结果会越来越大。

(5) 若两次输入 n 的值相差 1,但输出的结果却是一个正数,一个负数,那么两次输入的 n 可能是下面四组中的 ( C )。

A. 不可能      B. -6, -7      C. -15, -16      D. -23, -24

解析:

因为输入的整数 n,每次递归参数除以 2,所以只有相差 1 的负数才有可能符号不一样,每次在  $2^k$  处变符号,故应该选择 C。

(6) 此程序的时间复杂度是 ( B )。

A.  $O(n^2)$       B.  $O(\log n)$       C.  $O(n)$       D.  $O(n \log n)$

解析:

该程序每次调用,参数除以 2,快速收敛,类似于二分,复杂程度应该为  $\log n$ 。

2. 输入一串由小写字母组成的字符串,根据程序判断或选择正确的答案。

```

1  #include<stdio>
2  #include<string>
3  using namespace std;
4  int main() {
5      char str[60];
6      int len, i, j, chr[26];
7      char mmin='z';
8      scanf("%s",str);
9      len=strlen(str);

```

```

10     for(i= len-1;i>=1;i--)
11         if(str[i-1]<str[i])break;
12     if(i==0) {
13         printf("No result! \n");
14         return 0;
15     }
16     for (j=0;j<i-1;j++)putchar(str[j]);
17     memset(chr,0,sizeof(chr));
18     for(j=i;j<len;j++){
19         if(str[j]>str[i-1]&&str[j]<mmin)
20             mmin=str[j];
21         chr[str[j]-'a']++;
22     }
23     chr[mmin-'a']--;
24     chr[str[i-1]-'a']++;
25     putchar(mmin);
26     for(i=0;i<26;i++)
27         for(j=0;j<chr[i];j++)
28             putchar(i+'a');
29     putchar('\n') ;
30     return 0;
31 }

```

(1) 输入的字符串长度应该在 [1, 59] 的范围内。( )

答案 ✓

解析：字符串都是从 0 开始的，下标 0~59，共 60 个元素，但是字符串都有 “\0” 结尾，所以字符数是 [1, 59] 的范围内。

(2) 如果输入的字符数组所有字符都是从大到小的，那么会输出 “No result!”。( )

答案 ✕

解析：

```

for(i=len-1;i>=1;i--)
    if(str[i-1]<str[i])break;

```

前面的字符比后面的大，循环结束时 i 的值为 0，

```

if(i==0) {
    printf("No result! \n");
    return 0;
}

```

符合条件，输出 No result!

(3) 第 25 行输出的值为输入字符串里的 ASCII 最小的那个字符。( )

答案 ✕

解析：18 行到 22 行实际是要找出从 i 位置到最后比 str[i-1] 大的最小的字符。

(4) 第 26 行到第 28 行是把 chr[] 数组中存在的对应字符按照从小到大输出，即把剩下未输出的字符按照从小到大输出。( )

答案 ✓

解析：chr[]数组就相当于桶，按照从小到大存储了出现的字符的个数（不包括前面已经输出的字符）。

（5）如果输入的是abcdzdcba，则第16行输出的是（ A ）。

A. abc      B. abcd      C. abcdz      D. abcdzd

解析：这行代码是输出从后开始数第一个正序字符（即前一个字符比后一个小）之前的字母，故应为A。

（6）如果程序的输出结果是“ffghhghg”，则输入有可能是（ C ）。

A. ffghhghg      B. ffghhhgg      C. ffghghhg      D. ffghghgh

解析：

根据题意，应该从后往前，先找到第一个前面比后面小的字符，然后前面的字符一一输出，之后找出出现字符大的最小值输出，然后针对4个选项一一排查。

3. 本题是一款模拟贪吃蛇程序，游戏是在一个 $a \times a$ 的网格上进行的。其中输入第一行一个整数a。第二行两个整数n和m。接下来是n行，每行第一个数为opt，表示操作编号。接下来的输入的变量与操作编号对应，输出：即第m秒过后的地图，蛇所在的位置输出“o”，其余位置输出“.”，以换行结尾。

```
1  #include <bits/stdc++.h>
2  #include<windows.h>
3  using namespace std;
4  int a,mp[101][101];
5  int t[100003];
6  int y[100003];
7  int cnt;
8  int len=2,dir=3,die=0;
9  const int dx[5]={0,0,-1,0,1};
10 const int dy[5]={0,-1,0,1,0};
11 int nx=0,ny=1;
12 int px=1,py=2;
13 int check(int x,int yy)
14 {
15     if(x<1||x>a||yy<1||yy>a)
16         return 1;
17     if(cnt+1-mp[x][yy]<len)
18         return 1;
19     return 0;
20 }
21 void work()
22 {
23     if(die) return;
24     px+=nx;
25     py+=ny;
26     die=check(px,py);
27     if(die) return;
```

```

28     mp[px][py]=++cnt;
29 }
30 void show()
31 {
32     for(int i=1;i<=a;++i)
33     {
34         for(int j=1;j<=a;++j)
35         {
36             if(mp[i][j]!=0&&mp[i][j]>=cnt-len+1)
37                 putchar('o');
38             else
39                 putchar('.') ;
40         }
41         puts("");
42     }
43 }
44 int main()
45 {
46     mp[1][1]=++cnt;
47     mp[1][2]=++cnt;
48     int n,m,op,xx;
49     char s[3];
50     scanf("%d",&a);
51     scanf("%d%d",&n,&m);
52     while(n--)
53     {
54         scanf("%d%d",&op,&xx);
55         if(op==1)
56         {
57             t[xx]=1;
58             scanf("%s",s);
59             if(s[0]=='L')
60                 y[xx]=1;
61             else if(s[0]=='U')
62                 y[xx]=2;
63             else if(s[0]=='R')
64                 y[xx]=3;
65             else
66                 y[xx]=4;
67         }
68         else
69         {
70             t[xx]=2;
71         }
72     }
73     for(int tm=1;tm<=m;++tm)
74     {
75         if(t[tm]==1)
76         {
77             if(y[tm]%2!=dir%2)
78             {

```



```

79             dir=y[tm];
80             nx=dx[y[tm]];
81             ny=dy[y[tm]];
82         }
83     }
84     else if(t[tm]==2)
85     {
86         ++len;
87     }
88     work();
89     if(die)
90     {
91         break;
92     }
93 }
94 show();
95 return 0;
96 }

```

(1) (2 分) 由程序代码可知，贪吃蛇的初始长度为 2，蛇头和蛇尾分别在坐标 [1, 2]、[1, 1] 处。( )

答案 ✓

解析：由 46~47 行确定蛇身具体位置。

(2) (2 分) check 函数是用来检测蛇是否吃到果实的。( )

答案 ✕

解析：check 函数显然是用来检测蛇是否因越界致死。

(3) (2 分) 第 54 行及第 58 行输入 1 x y 表示在第 x 秒按下了 y 键，y 为 LURD 中的一种，分别表示按下了左、上、右、下四种按钮。( )

答案 ✓

解析：简单模拟如 58~71 行所示。

(4) (2 分) 当输入样例如下所示时：

```

10
10 20
2 1
2 2
2 3
2 4
2 5
1 6 R
1 7 D
1 8 L
1 9 U

```

最终程序的运行结果所代表的含义可表示为贪吃蛇在第 9 秒过后就死亡了，因此最后贪吃蛇保持的是死亡前（第 7 秒过后）的位置。( )

答案 ✕

解析：本题本质就是个模拟题，只要一步一步代入进程序规划即可得解，可表示为：贪吃蛇在第 9 秒后就死亡了，因此最后贪吃蛇保持的是死亡前（第 8 秒过后）。

（5）若输入地图边长为  $x$ ，共  $n$  次操作 ( $x > n$ )，则该程序时间复杂度为（ ）。

A.  $x^2$     B.  $n^2$     C.  $n^2 * x$     D.  $x^2 * n$

解析：这道题主要的时间都是用在模拟贪吃蛇的运动以及最后的绘图之上，很容易得到最后的渐进时间复杂度为  $O(x^2)$ 。

完善程序

1.（APP 点餐）小 D 在外玩，回来时想在 APP 上点 KFC，然后回宿舍。他想选择一家 KFC，取了食品后尽快回宿舍，忽略取餐时间。地图可看作是  $n * m$  的网格，其中有一些不可通过的障碍，小 D、KFC、宿舍均可以看做是道路，可以通过，他可穿过多个 KFC。小 D 可以选择上下左右四个方向移动，一次移动算一步。求他取到餐并回到宿舍的最小步数。

输入第一行有两个数  $n, m$ 。

接下来  $n$  行，每行包含  $m$  个字符，表示地图。

“S”表示小 D 初始位置，

“E”表示宿舍位置

“#”表示障碍物

“.”表示道路

“K”表示 KFC

下面的程序已采用宽度优先搜索的算法完成这个问题，试补全程序。

```
1  #include<bits/stdc++.h>
2  #define fi first
3  #define se second
4  using namespace std;
5  const int MAXN=1e3+10;
6  const int INF=0x3f3f3f3f;
7  typedef pair<int,int>P;
8  char s[MAXN][MAXN];
9  int n,m;
10 int dir[2][4]={1,-1,0,0},{0,0,1,-1}};
11 int dis[2][MAXN][MAXN];
12
13 void bfs(int p,int a,int b){
14     memset(dis[p],INF,sizeof(dis[p]));
15     dis[p][a][b]=0;
16     queue<pair<P,int>>q;
17     q.push({a,b,0});
18     while(!q.empty()){
19         int x=q.front().fi.fi,
20         y=q.front().fi.se,
21         d=q.front().se;
22         q.pop();
23         for(int i=0;i<4;i++){
```

```

24         int dx=x+dir[0][i],
25         dy=y+dir[1][i];
26         if(dx<1||dy<1||dx>n||dy>m||s[dx][dy]=='#')
27             continue;
28         if(____①____){
29             ____②____;
30             ____③____;
31         }
32     }
33 }
34 }
35
36 int main(void)
37 {
38     while(scanf("%d%d",&n,&m)!=EOF){
39         int ans=INF;
40         scanf("%d%d",&n,&m);
41         for(int i=1;i<=n;i++)
42             scanf("%s",s[i]+1);
43         for(int i=1;i<=n;i++)
44             for(int j=1;j<=m;j++)
45                 if(s[i][j]=='S')
46                     bfs(0,i,j);
47                 else if(s[i][j]=='E')
48                     ____④____;
49         for(int i=1;i<=n;i++)
50             for(int j=1;j<=m;j++)
51                 if(s[i][j]=='K')
52                     ans=⑤;
53         if(ans==INF)
54             ans=-1;
55         printf("%d\n",ans);
56     }
57     return 0;
58 }

```

(1) ①处应填 ( B )。

- A. `dis[p][dx][dy]>d`      B. `dis[p][dx][dy]>d+1`  
C. `dis[p][dx][dy]<d`      D. `dis[p][dx][dy]<d+1`

解析：找最短路径，发现有一条到`[dx][dy]`点更短的路，因为走一格需要1个步长，所以要+1。

(2) ②处应填 ( C )。

- A. `dis[p][dx][dy]=d`      B. `dis[p][dx][dy]=d-1`  
C. `dis[p][dx][dy]=d+1`      D. `dis[p][dx][dy]=1`

解析：

发现有更短的路就更新。

(3) ③处应填 ( A )。

- A. `q.push({dx,dy},d+1)`      B. `q.push({dx,dy},d)`

C. q.push({dx, dy}, d-1)      D. q.push({dx, dy}, 1)

解析：发现更短路径后更新该点路径，还需要加入队列，从而可以松弛更新后续结点的路径值。

(4) ④处应填 ( D )。

A. bfs(0, j, i)      B. bfs(1, j, i)

C. bfs(0, i, j)      D. bfs(1, i, j)

解析：搜索时的第一个参数为 0，是以小 D 所在位置作为源点进行搜索，第一个参数为 1，则以宿舍作为源点进行搜索。

(5) ⑤处应填 ( A )。

A. min(ans, dis[0][i][j]+dis[1][i][j])

B. min(ans, dis[0][j][i]+dis[1][j][i])

C. min(ans, dis[0][i][j])

D. min(ans, dis[1][i][j])

解析：

很显然，我们需要的是从小 D 位置到该 KFC 所在点及从宿舍位置到该 KFC 所在点位置距离和的最小值，故应该选择 A。

2. (尺取法求区间个数) 给 n 个非负整数  $a[1], a[2], \dots, a[n]$ , 求区间和小于或等于 k 的区间个数, 即求使  $SUM = a[L] + a[L+1] + \dots + a[R-1] + a[R] \leq k$  的区间  $[L, R]$  的个数 ( $1 \leq L \leq R \leq n$ ), 但由于对内存和复杂度有要求, 本题已经用尺取法写好部分代码, 请补全程序。

输入第一行两个整数 n, k ( $1 \leq n \leq 1000000, 0 \leq k \leq 10000000000000000$ )。第二行为 n 个数, 表示  $a[1] \sim a[n]$  的值 ( $0 \leq a[i] \leq 1000000000$ )。

```
1  #include<bits/stdc++.h>
2  # define ll long long
3  using namespace std;
4  const int mx=1e6+10;
5  int n,a[mx];
6  ll k,sum,ans;
7  int main()
8  {
9      scanf("%d%lld",&n,&k);
10     for(int i=1;i<=n;++i)
11     {
12         scanf("%d",&a[i]);
13     }
14     int r=0;
15     for (int i=1;i<=n;++i)
16     {
17         while(r<n)
18         {
19             if(____①____) ____②____;
20             else break;
21         }
```

```

22      ____③____;
23      if(i<=r) ____ ④____;
24      else ____⑤____;
25  }
26  printf("%lld\n",ans);
27  }

```

(1) ①处应填 ( A )。

- A.  $\text{sum}+\text{a}[\text{r}+1]<=k$     B.  $\text{sum}+\text{a}[\text{r}]<=k$   
 C.  $\text{sum}+\text{a}[\text{r}+1]<k$     D.  $\text{sum}+\text{a}[\text{r}]<k$

解析：根据题意求 $\leq k$ ，因此只有 A,B 两个选项符合，首先 r 的初始值为 0，二 a 数组是从 1 开始的。所以仅从这一点上 B 也不正确。

(2) ②处应填 ( B )。

- A.  $\text{sum}+=\text{a}[\text{r}]$     B.  $\text{sum}+=\text{a}[++\text{r}]$   
 C.  $\text{sum}+=\text{a}[\text{r}++]$     D.  $\text{sum}+=\text{a}[\text{r}+1]$

解析：当符合  $\text{sum}+\text{a}[\text{r}+1]<=k$  这个条件时，就要把 r+1 这个加入 sum，并且 r 移动到 r+1 下标，所以 r 还要自增，简写就是  $\text{a}[++\text{r}]$ 。

(3) ③处应填 ( A )。

- A.  $\text{ans}+=\text{r}-\text{i}+1$     B.  $\text{ans}+=\text{r}-\text{i}$   
 C.  $\text{ans}+=\text{r}-\text{i}-1$     D.  $\text{ans}+=\text{n}-\text{i}+1$

解析：如果 sum 加上下一个数不符合 $\leq k$ 的条件，终止 while 循环，求出 i 到 r 的数组元素个数。

(4) ④处应填 ( D )。

- A.  $\text{sum}+=\text{a}[\text{i}]$     B.  $\text{sum}=\text{a}[\text{r}]$   
 C.  $\text{sum}=\text{a}[\text{i}]$     D.  $\text{sum}-=\text{a}[\text{i}]$

解析：本题采用尺取法同向扫描快慢指针，if( $\text{i}<=\text{r}$ ) 的时候左端点右移一步，并且左端点的值也在 sum 中减去，再加上 r+1 位置的值，判断是否符合条件。如果尺取法不熟练的同学，可以代入一组数模拟一遍。即能选出正确选项。

(5) ⑤处应填 ( D )。

- A.  $\text{r}++\text{i}$     B.  $\text{r}=\text{i}--$   
 C.  $\text{r}=\text{i}++$     D.  $\text{r}=\text{i}$

解析：当数据中有一个数比较大，例如大于 k 的数，任何值加上他都会大于 k 那么这个时候需要把 i 赋值给 r，这样下次  $\text{sum}+\text{a}[\text{r}+1]$ ，继续计算符合条件的值。

如果还是不太明白，模拟下面数据即可。

5 8

3 1 4 10 2