2021 CCF 非专业级别软件能力认证第一轮

（CSP-S1）提高级 C++语言试题

认证时间：2021 年 9 月 19 日 09:30~11:30

考生注意事项：

- 试题纸共有 16 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

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. 在 Linux 系统终端中，用于列出当前目录下所含的文件和子目录的命令为（ ）。

A. ls

B. cd

C. cp

D. all

2. 二进制数 00101010_2 和 00010110_2 的和为（ ）。

A. 00111100_2

B. 01000000_2

C. 00111100_2

D. 01000010_2

3. 在程序运行过程中，如果递归调用的层数过多，可能会由于（ ）引发错误。

A. 系统分配的栈空间溢出

B. 系统分配的队列空间溢出

C. 系统分配的链表空间溢出

D. 系统分配的堆空间溢出

4. 以下排序方法中，（ ）是不稳定的。

A. 插入排序

B. 冒泡排序

C．堆排序

D．归并排序

5．以比较为基本运算，对于 2n 个数，同时找到最大值和最小值，最坏情况下需要的最小的比较次数为（　）。

A．4n-2

B．3n+1

C．3n-2

D．2n+1

6．现有一个地址区间为 0～10 的哈希表，对于出现冲突情况，会往后找第一个空的地址存储（到 10 冲突了就从 0 开始往后），现在要依次存储（0，1，2，3，4，5，6，7），哈希函数为 $h(x)=x^2 \bmod 11$。请问 7 存储在哈希表哪个地址中（　）。

A．5

B．6

C．7

D．8

7．G 是一个非连通简单无向图（没有自环和重边），共有 36 条边，则该图至少有（　）个点。

A．8

B．9

C．10

D．11

8．令根结点的高度为 1，则一棵含有 2021 个结点的二叉树的高度至少为（　）。

A．10

B．11

C．12

D．2021

9. 前序遍历和中序遍历相同的二叉树为且仅为（　）。

A. 只有 1 个点的二叉树

B. 根结点没有左子树的二叉树

C. 非叶子结点只有左子树的二叉树

D. 非叶子结点只有右子树的二叉树

10. 定义一种字符串操作为交换相邻两个字符。将“DACFEB”变为“ABCDEF”最少需要（　）次上述操作。

A. 7

B. 8

C. 9

D. 6

11. 有如下递归代码

```
solve(t, n):
  if t=1 return 1
  else return 5*solve(t-1,n) mod n
```

则 solve(23,23)的结果为（　）。

A. 1

B. 7

C. 12

D. 22

12. 斐波那契数列的定义为：$F_1=1$，$F_2=1$，$F_n=F_{n-1}+F_{n-2}$ (n>=3)。现在用如下程序来计算斐波那契数列的第 n 项，其时间复杂度为（　）。

```
F(n):
  if n<=2 return 1
  else return F(n-1) + F(n-2)
```

A. $O(n)$

B. $O(n^2)$

C. $O(2^n)$

D. $O(n \log n)$

13. 有 8 个苹果从左到右排成一排，你要从中挑选至少一个苹果，并且不能同时挑选相邻的两个苹果，一共有（ ）种方案。

A. 36

B. 48

C. 54

D. 64

14. 设一个三位数 n= $\overline{abc}$，a，b，c 均为 1～9 之间的整数，若以 a、 b、 c 作为三角形的三条边可以构成等腰三角形（包括等边），则这样的 n 有（ ）个。

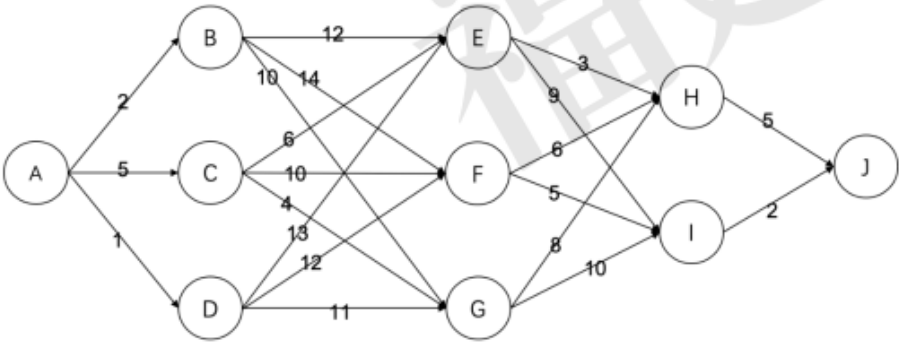
A. 81

B. 120

C. 165

D. 216

15. 有如下的有向图，节点为 A，B，…，J，其中每条边的长度都标在图中。则节点 A 到节点 J 的最短路径长度为（ ）。



A. 16

B. 19

C. 20

D. 22

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)

```
01 #include <iostream>
02 #include <cmath>
03 using namespace std;
04
05 const double r = acos(0.5);
06
07 int a1, b1, c1, d1;
08 int a2, b2, c2, d2;
09
10 inline int sq(const int x) { return x * x; }
11 inline int cu(const int x) { return x * x * x; }
12
13 int main()
14 {
15     cout.flags(ios::fixed);
16     cout.precision(4);
17
18     cin >> a1 >> b1 >> c1 >> d1;
19     cin >> a2 >> b2 >> c2 >> d2;
20
21     int t = sq(a1 - a2) + sq(b1 - b2) + sq(c1 - c2);
22
23     if (t <= sq(d2 - d1)) cout << cu(min(d1, d2)) * r * 4;
24     else if (t >= sq(d2 + d1)) cout << 0;
25     else {
26         double x = d1 - (sq(d1) - sq(d2) + t) / sqrt(t) / 2;
27         double y = d2 - (sq(d2) - sq(d1) + t) / sqrt(t) / 2;
28         cout << (x * x * (3 * d1 - x) + y * y * (3 * d2 - y)) * r;
29     }
30     cout << endl;
31     return 0;
32 }
```

假设输入的所有数的绝对值都不超过 1000，完成下面的判断题和单选题：

- 判断题

16. 将第 21 行中 t 的类型声明从 int 改为 double，**不会**影响程序运行的结果。（ ）

17. 将第 26、27 行中的“/ sqrt(t) / 2”替换为“/ 2 / sqrt(t)”，**不会**影响程序运行的结果。（ ）

18. 将第 28 行中的“x * x”改成“sq(x)”、“y * y”改成“sq(y)” ，**不会**影响程序运行的结果。（ ）

19. **（2 分）**当输入为“0 0 0 1 1 0 0 1”时，输出为“1.3090”。（ ）

- 单选题

20. 当输入为“1 1 1 1 1 1 1 2”时，输出为（ ）。

A. “3.1416” B. “6.2832” C. “4.7124” D. “4.1888”

21. **（2.5 分）**这段代码的含义为（ ）。

A. 求圆的面积并　　B. 求球的体积并
C. 求球的体积交　　D. 求椭球的体积并

(2)

```
01 #include <algorithm>
02 #include <iostream>
03 using namespace std;
04
05 int n, a[1005];
06
07 struct Node
08 {
09     int h, j, m, w;
10
11     Node(const int _h, const int _j, const int _m, const int _w):
12         h(_h), j(_j), m(_m), w(_w)
13     { }
14
15     Node operator+(const Node &o) const
16     {
17         return Node(
18             max(h, w + o.h),
19             max(max(j, o.j), m + o.h),
20             max(m + o.w, o.m),
21             w + o.w);
22     }
```

```
23 };
24
25 Node solve1(int h, int m)
26 {
27     if (h > m)
28         return Node(-1, -1, -1, -1);
29     if (h == m)
30         return Node(max(a[h], 0), max(a[h], 0), max(a[h], 0), a[h]);
31     int j = (h + m) >> 1;
32     return solve1(h, j) + solve1(j + 1, m);
33 }
34
35 int solve2(int h, int m)
36 {
37     if (h > m)
38         return -1;
39     if (h == m)
40         return max(a[h], 0);
41     int j = (h + m) >> 1;
42     int wh = 0, wm = 0;
43     int wht = 0, wmt = 0;
44     for (int i = j; i >= h; i--) {
45         wht += a[i];
46         wh = max(wh, wht);
47     }
48     for (int i = j + 1; i <= m; i++) {
49         wmt += a[i];
50         wm = max(wm, wmt);
51     }
52     return max(max(solve2(h, j), solve2(j + 1, m)), wh + wm);
53 }
54
55 int main()
56 {
57     cin >> n;
58     for (int i = 1; i <= n; i++) cin >> a[i];
59     cout << solve1(1, n).j << endl;
60     cout << solve2(1, n) << endl;
61     return 0;
62 }
```

假设输入的所有数的绝对值都不超过 1000，完成下面的判断题和单选题：

- 判断题

22.程序**总是**会正常执行并输出两行两个相等的数。（　）

23. 第 28 行与第 38 行分别有可能执行两次及以上。（ ）

24. 当输入为“5 -10 11 -9 5 -7”时，输出的第二行为“7”。（ ）

- 单选题

25. solve1(1, n) 的时间复杂度为（ ）。

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

26. solve2(1, n) 的时间复杂度为（ ）。

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

27. 当输入为“10 -3 2 10 0 -8 9 -4 -5 9 4”时，输出的第一行为（ ）。

A. “13”　　B. “17”　　C. “24”　　D. “12”

(3)

```
01 #include <iostream>
02 #include <string>
03 using namespace std;
04
05 char base[64];
06 char table[256];
07
08 void init()
09 {
10     for (int i = 0; i < 26; i++) base[i] = 'A' + i;
11     for (int i = 0; i < 26; i++) base[26 + i] = 'a' + i;
12     for (int i = 0; i < 10; i++) base[52 + i] = '0' + i;
13     base[62] = '+', base[63] = '/';
14
15     for (int i = 0; i < 256; i++) table[i] = 0xff;
16     for (int i = 0; i < 64; i++) table[base[i]] = i;
17     table['='] = 0;
18 }
19
20 string encode(string str)
21 {
22     string ret;
23     int i;
24     for (i = 0; i + 3 <= str.size(); i += 3) {
25         ret += base[str[i] >> 2];
26         ret += base[(str[i] & 0x03) << 4 | str[i + 1] >> 4];
27         ret += base[(str[i + 1] & 0x0f) << 2 | str[i + 2] >> 6];
28         ret += base[str[i + 2] & 0x3f];
```

```
29      }
30      if (i < str.size()) {
31          ret += base[str[i] >> 2];
32          if (i + 1 == str.size()) {
33              ret += base[(str[i] & 0x03) << 4];
34              ret += "==";
35          }
36          else {
37              ret += base[(str[i] & 0x03) << 4 | str[i + 1] >> 4];
38              ret += base[(str[i + 1] & 0x0f) << 2];
39              ret += "=";
40          }
41      }
42      return ret;
43  }
44
45  string decode(string str)
46  {
47      string ret;
48      int i;
49      for (i = 0; i < str.size(); i += 4) {
50          ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
51          if (str[i + 2] != '=')
52              ret += (table[str[i + 1]] & 0x0f) << 4 | table[str[i +
                                                          2]] >> 2;
53          if (str[i + 3] != '=')
54              ret += table[str[i + 2]] << 6 | table[str[i + 3]];
55      }
56      return ret;
57  }
58
59  int main()
60  {
61      init();
62      cout << int(table[0]) << endl;
63
64      int opt;
65      string str;
66      cin >> opt >> str;
67      cout << (opt ? decode(str) : encode(str)) << endl;
68      return 0;
69  }
```

假设输入总是合法的（一个整数和一个不含空白字符的字符串，用空格隔开），完成下面的判断题和单选题：

- 判断题

28. 程序总是先输出**一行**一个整数，再输出**一行**一个字符串。（ ）

29. 对于任意不含空白字符的字符串 str1，先执行程序输入“0 str1”，得到输出的第二行记为 str2；再执行程序输入“1 str2”，输出的第二行必为 str1。（ ）

30. 当输入为“1 SGVsbG93b3JsZA==”时，输出的第二行为“HelloWorld”。（ ）

- 单选题

31. 设输入字符串长度为 n，encode 函数的时间复杂度为（ ）。

A. $\Theta(\sqrt{n})$　　B. $\Theta(n)$　　C. $\Theta(n\log n)$　　D. $\Theta(n^2)$

32. 输出的第一行为（ ）。

A. “0xff”　　B. “255”　　C. “0xFF”　　D. “-1”

33. **（4 分）**当输入为“0 CSP2021csp”时，输出的第二行为（ ）。

A. “Q1NQMjAyMWNzcAv=”　　B. “Q1NQMjAyMGNzcA==”

C. “Q1NQMjAyMGNzcAv=”　　D. “Q1NQMjAyMWNzcA==”

三、 完善程序（单选题，每小题 3 分，共计 30 分）

(1)（魔法数字）小 H 的魔法数字是 4。给定 n，他希望用若干个 4 进行若干次加法、减法和整除运算得到 n。但由于小 H 计算能力有限，计算过程中只能出现不超过 $M = 10000$ 的正整数。求至少可能用到多少个 4。

例如，当 $n = 2$ 时，有 $2 = (4 + 4)/4$，用到了 3 个 4，是最优方案。

试补全程序。

```
01 #include <iostream>
02 #include <cstdlib>
03 #include <climits>
04
05 using namespace std;
06
07 const int M = 10000;
08 bool Vis[M + 1];
09 int F[M + 1];
10
```

```
11 void update(int &x, int y) {
12     if (y < x)
13         x = y;
14 }
15
16 int main() {
17     int n;
18     cin >> n;
19     for (int i = 0; i <= M; i++)
20         F[i] = INT_MAX;
21     ①;
22     int r = 0;
23     while (②) {
24         r++;
25         int x = 0;
26         for (int i = 1; i <= M; i++)
27             if (③)
28                 x = i;
29         Vis[x] = 1;
30         for (int i = 1; i <= M; i++)
31             if (④) {
32                 int t = F[i] + F[x];
33                 if (i + x <= M)
34                     update(F[i + x], t);
35                 if (i != x)
36                     update(F[abs(i - x)], t);
37                 if (i % x == 0)
38                     update(F[i / x], t);
39                 if (x % i == 0)
40                     update(F[x / i], t);
41             }
42     }
43     cout << F[n] << endl;
44     return 0;
45 }
```

34. ①处应填（　）

A. `F[4] = 0`　　B. `F[1] = 4`　　C. `F[1] = 2`　　D. `F[4] = 1`

35. ②处应填（　）

A. `!Vis[n]`　　B. `r < n`

C. `F[M] == INT_MAX`　　D. `F[n] == INT_MAX`

36. ③处应填（　）

A. `F[i] == r`　　B. `!Vis[i] && F[i] == r`

C. `F[i] < F[x]`　　D. `!Vis[i] && F[i] < F[x]`

37. ④处应填（　）

A. `F[i] < F[x]`　B. `F[i] <= r`　C. `Vis[i]`　D. `i <= x`

(2)（RMQ 区间最值问题）给定序列 $a_0,\dots,a_{n-1}$，和 m 次询问，每次询问给定 l,r，求 $\max\{a_l,\dots,a_r\}$。

为了解决该问题，有一个算法叫 the Method of Four Russians，其时间复杂度为 $O(n+m)$，步骤如下：

- 建立 Cartesian（笛卡尔）树，将问题转化为树上的 LCA（最近公共祖先）问题。
- 对于 LCA 问题，可以考虑其 Euler 序（即按照 DFS 过程，经过所有点，环游回根的序列），即求 Euler 序列上两点间**一个新的 RMQ 问题。**
- 注意新的问题为 ±1 RMQ，即相邻两点的深度差一定为 1。

下面解决这个 ±1 RMQ 问题，“序列”指 Euler 序列：

- 设 t 为 Euler 序列长度。取 $b=\left\lceil\frac{\log_2 t}{2}\right\rceil$。将序列每 b 个分为一大块，使用 ST 表（倍增表）处理大块间的 RMQ 问题，复杂度 $O\left(\frac{t}{b}\log t\right)=O(n)$。
- **（重点）**对于一个块内的 RMQ 问题，也需要$O(1)$ 的算法。由于差分数组 2^{b-1} 种，可以预处理出所有情况下的最值位置，预处理复杂度 $O(b2^b)$，不超过 $O(n)$。
- 最终，对于一个查询，可以转化为中间整的大块的 RMQ 问题，以及两端块内的 RMQ 问题。

试补全程序。

```
001 #include <iostream>
002 #include <cmath>
003
004 using namespace std;
005
006 const int MAXN = 100000, MAXT = MAXN << 1;
007 const int MAXL = 18, MAXB = 9, MAXC = MAXT / MAXB;
008
009 struct node {
010     int val;
011     int dep, dfn, end;
```

```
012     node *son[2]; // son[0], son[1] 分别表示左右儿子
013 } T[MAXN];
014
015 int n, t, b, c, Log2[MAXC + 1];
016 int Pos[(1 << (MAXB - 1)) + 5], Dif[MAXC + 1];
017 node *root, *A[MAXT], *Min[MAXL][MAXC];
018
019 void build() { // 建立 Cartesian 树
020     static node *S[MAXN + 1];
021     int top = 0;
022     for (int i = 0; i < n; i++) {
023         node *p = &T[i];
024         while (top && S[top]->val < p->val)
025             ①;
026         if (top)
027             ②;
028         S[++top] = p;
029     }
030     root = S[1];
031 }
032
033 void DFS(node *p) { // 构建 Euler 序列
034     A[p->dfn = t++] = p;
035     for (int i = 0; i < 2; i++)
036         if (p->son[i]) {
037             p->son[i]->dep = p->dep + 1;
038             DFS(p->son[i]);
039             A[t++] = p;
040         }
041     p->end = t - 1;
042 }
043
044 node *min(node *x, node *y) {
045     return ③ ? x : y;
046 }
047
048 void ST_init() {
049     b = (int)(ceil(log2(t) / 2));
050     c = t / b;
051     Log2[1] = 0;
052     for (int i = 2; i <= c; i++)
053         Log2[i] = Log2[i >> 1] + 1;
054     for (int i = 0; i < c; i++) {
055         Min[0][i] = A[i * b];
```

```
056         for (int j = 1; j < b; j++)
057             Min[0][i] = min(Min[0][i], A[i * b + j]);
058     }
059     for (int i = 1, l = 2; l <= c; i++, l <<= 1)
060         for (int j = 0; j + l <= c; j++)
061             Min[i][j] = min(Min[i - 1][j], Min[i - 1][j + (l >>
                                                          1)]);
062 }
063
064 void small_init() { // 块内预处理
065     for (int i = 0; i <= c; i++)
066         for (int j = 1; j < b && i * b + j < t; j++)
067             if (④)
068                 Dif[i] |= 1 << (j - 1);
069     for (int S = 0; S < (1 << (b - 1)); S++) {
070         int mx = 0, v = 0;
071         for (int i = 1; i < b; i++) {
072             ⑤;
073             if (v < mx) {
074                 mx = v;
075                 Pos[S] = i;
076             }
077         }
078     }
079 }
080
081 node *ST_query(int l, int r) {
082     int g = Log2[r - l + 1];
083     return min(Min[g][l], Min[g][r - (1 << g) + 1]);
084 }
085
086 node *small_query(int l, int r) { // 块内查询
087     int p = l / b;
088     int S = ⑥;
089     return A[l + Pos[S]];
090 }
091
092 node *query(int l, int r) {
093     if (l > r)
094         return query(r, l);
095     int pl = l / b, pr = r / b;
096     if (pl == pr) {
097         return small_query(l, r);
098     } else {
```

```
099         node *s = min(small_query(l, pl * b + b - 1),
                                      small_query(pr * b, r));
100         if (pl + 1 <= pr - 1)
101             s = min(s, ST_query(pl + 1, pr - 1));
102         return s;
103     }
104 }
105
106 int main() {
107     int m;
108     cin >> n >> m;
109     for (int i = 0; i < n; i++)
110         cin >> T[i].val;
111     build();
112     DFS(root);
113     ST_init();
114     small_init();
115     while (m--) {
116         int l, r;
117         cin >> l >> r;
118         cout << query(T[l].dfn, T[r].dfn)->val << endl;
119     }
120     return 0;
121 }
```

38. ①处应填（ ）

A. `p->son[0] = S[top--]`
B. `p->son[1] = S[top--]`
C. `S[top--]->son[0] = p`
D. `S[top--]->son[1] = p`

39. ②处应填（ ）

A. `p->son[0] = S[top]`
B. `p->son[1] = S[top]`
C. `S[top]->son[0] = p`
D. `S[top]->son[1] = p`

40. ③处应填（ ）

A. `x->dep < y->dep`
B. `x < y`
C. `x->dep > y->dep`
D. `x->val < y->val`

41. ④处应填（ ）

A. `A[i * b + j - 1] == A[i * b + j]->son[0]`
B. `A[i * b + j]->val < A[i * b + j - 1]->val`
C. `A[i * b + j] == A[i * b + j - 1]->son[1]`
D. `A[i * b + j]->dep < A[i * b + j - 1]->dep`

42. ⑤处应填（　）

A. `v += (S >> i & 1) ? -1 : 1`
B. `v += (S >> i & 1) ? 1 : -1`
C. `v += (S >> (i - 1) & 1) ? 1 : -1`
D. `v += (S >> (i - 1) & 1) ? -1 : 1`

43. ⑥处应填（　）

A. `(Dif[p] >> (r - p * b)) & ((1 << (r - l)) - 1)`
B. `Dif[p]`
C. `(Dif[p] >> (l - p * b)) & ((1 << (r - l)) - 1)`
D. `(Dif[p] >> ((p + 1) * b - r)) & ((1 << (r - l + 1)) - 1)`

一、单项选择题（共 15 题，每题 2 分，共计 30 分）

1	2	3	4	5	6	7	8	9	10
A	B	A	C	C	C	C	B	D	A
11	12	13	14	15					
A	C	C	C	B					

二、阅读程序（除特殊说明外，判断题 1.5 分，单选题 3 分，共计 40 分）

第 1 题	判断题（填√或×）				单选题	
	16)	17)	18)	19)(2 分)	20)	21) (2.5 分)
	√	×	×	√	D	C
第 2 题	判断题（填√或×）			单选题		
	22)	23)	24)	25)	26)	27)
	√	×	×	B	C	B
第 3 题	判断题（填√或×）			单选题		
	28)	29)	30)	31)	32)	33) (4 分)
	×	√	×	B	D	D

三、完善程序（单选题，每小题 3 分，共计 30 分）

第 1 题				第 2 题					
34)	35)	36)	37)	38)	39)	40)	41)	42)	43)
D	A	D	C	A	D	A	D	D	C

2021 CCF 非专业级别软件能力认证第一轮

（CSP-J1）入门级 C++语言试题

认证时间：2021 年 9 月 19 日 14:30~16:30

考生注意事项：

- 试题纸共有 12 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

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．以下不属于面向对象程序设计语言的是（ ）。

A．C++

B．Python

C．Java

D．C

2．以下奖项与计算机领域最相关的是（ ）。

A．奥斯卡奖

B．图灵奖

C．诺贝尔奖

D．普利策奖

3．目前主流的计算机储存数据最终都是转换成（ ）数据进行储存。

A．二进制

B．十进制

C．八进制

D．十六进制

4．以比较作为基本运算，在 N 个数中找出最大数，最坏情况下所需要的最少的比较次数为（ ）。

A．N^2

B. N

C. N-1

D. N+1

5. 对于入栈顺序为 a, b, c, d, e 的序列，下列（ ）不是合法的出栈序列。

A. a, b, c, d, e

B. e, d, c, b, a

C. b, a, c, d, e

D. c, d, a, e, b

6. 对于有 n 个顶点、m 条边的无向连通图 (m>n)，需要删掉（ ）条边才能使其成为一棵树。

A. n-1

B. m-n

C. m-n-1

D. m-n+1

7. 二进制数 101.11 对应的十进制数是（ ）。

A. 6.5

B. 5.5

C. 5.75

D. 5.25

8. 如果一棵二叉树只有根结点，那么这棵二叉树高度为 1。请问高度为 5 的完全二叉树有（ ）种不同的形态？

A. 16

B. 15

C. 17

D. 32

9. 表达式 a*(b+c)*d 的后缀表达式为(　)，其中“*”和“+”是运算符。

A. **a+bcd

B. abc+*d*

C. abc+d**

D. *a*+bcd

10. 6 个人，两个人组一队，总共组成三队，不区分队伍的编号。不同的组队情况有（　）种。

A. 10

B. 15

C. 30

D. 20

11. 在数据压缩编码中的哈夫曼编码方法，在本质上是一种（　）的策略。

A. 枚举

B. 贪心

C. 递归

D. 动态规划

12. 由 1，1，2，2，3 这五个数字组成不同的三位数有（　）种。

A. 18

B. 15

C. 12

D. 24

13. 考虑如下递归算法

```
solve(n)
  if n<=1 return 1
```

```
    else if n>=5 return n*solve(n-2)
    else return n*solve(n-1)
```

则调用 solve(7)得到的返回结果为（　）。

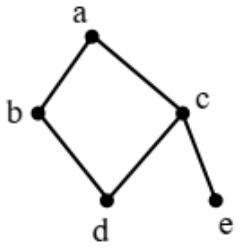
A. 105

B. 840

C. 210

D. 420

14. 以 a 为起点，对右边的无向图进行深度优先遍历，则 b、 c、 d、 e 四个点中有可能作为最后一个遍历到的点的个数为（　）。



A. 1

B. 2

C. 3

D. 4

15. 有四个人要从 A 点坐一条船过河到 B 点，船一开始在 A 点。该船一次最多可坐两个人。已知这四个人中每个人独自坐船的过河时间分别为 1，2，4，8，且两个人坐船的过河时间为两人独自过河时间的较大者。则最短（　）时间可以让四个人都过河到 B 点（包括从 B 点把船开回 A 点的时间）。

A. 14

B. 15

C. 16

D. 17

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)

```
01 #include <iostream>
02 using namespace std;
03
04 int n;
```

```
05 int a[1000];
06
07 int f(int x)
08 {
09     int ret = 0;
10     for (; x; x &= x - 1) ret++;
11     return ret;
12 }
13
14 int g(int x)
15 {
16     return x & -x;
17 }
18
19 int main()
20 {
21     cin >> n;
22     for (int i = 0; i < n; i++) cin >> a[i];
23     for (int i = 0; i < n; i++)
24         cout << f(a[i]) + g(a[i]) << ' ';
25     cout << endl;
26     return 0;
27 }
```

- 判断题

16. 输入的 n 等于 1001 时，程序**不会**发生下标越界。（ ）

17. 输入的 a[i] 必须全为正整数，否则程序将陷入死循环。（ ）

18. 当输入为“5 2 11 9 16 10”时，输出为“3 4 3 17 5”。（ ）

19. 当输入为“1 511998”时，输出为“18”。（ ）

20. 将源代码中 g 函数的定义（14-17 行）移到 main 函数的后面，程序可以正常编译运行。（ ）

- 单选题

21. 当输入为“2 -65536 2147483647”时，输出为（ ）。

A. “65532 33” B. “65552 32” C. “65535 34” D. “65554 33”

（2）

```
01 #include <iostream>
02 #include <string>
```

```
03 using namespace std;
04
05 char base[64];
06 char table[256];
07
08 void init()
09 {
10     for (int i = 0; i < 26; i++) base[i] = 'A' + i;
11     for (int i = 0; i < 26; i++) base[26 + i] = 'a' + i;
12     for (int i = 0; i < 10; i++) base[52 + i] = '0' + i;
13     base[62] = '+', base[63] = '/';
14
15     for (int i = 0; i < 256; i++) table[i] = 0xff;
16     for (int i = 0; i < 64; i++) table[base[i]] = i;
17     table['='] = 0;
18 }
19
20 string decode(string str)
21 {
22     string ret;
23     int i;
24     for (i = 0; i < str.size(); i += 4) {
25         ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
26         if (str[i + 2] != '=')
27             ret += (table[str[i + 1]] & 0x0f) << 4 | table[str[i +
                                                              2]] >> 2;
28         if (str[i + 3] != '=')
29             ret += table[str[i + 2]] << 6 | table[str[i + 3]];
30     }
31     return ret;
32 }
33
34 int main()
35 {
36     init();
37     cout << int(table[0]) << endl;
38
39     string str;
40     cin >> str;
41     cout << decode(str) << endl;
42     return 0;
43 }
```

- 判断题

22. 输出的第二行一定是由小写字母、大写字母、数字和“+”、“/”、“=”构成的字符串。（ ）

23. 可能存在输入不同，但输出的第二行相同的情形。（ ）

24. 输出的第一行为“-1”。（ ）

- 单选题

25. 设输入字符串长度为 n，decode 函数的时间复杂度为（ ）。

A. $\Theta(\sqrt{n})$　　B. $\Theta(n)$　　C. $\Theta(n \log n)$　　D. $\Theta(n^2)$

26. 当输入为“Y3Nx”时，输出的第二行为（ ）。

A. “csp”　　B. “csq”　　C. “CSP”　　D. “Csp”

27. **（3.5 分）**当输入为“Y2NmIDIwMjE=”时，输出的第二行为（ ）。

A. “ccf2021”　　B. “ccf2022”　　C. “ccf 2021”　　D. “ccf 2022”

（3）

```
01 #include <iostream>
02 using namespace std;
03
04 const int n = 100000;
05 const int N = n + 1;
06
07 int m;
08 int a[N], b[N], c[N], d[N];
09 int f[N], g[N];
10
11 void init()
12 {
13     f[1] = g[1] = 1;
14     for (int i = 2; i <= n; i++) {
15         if (!a[i]) {
16             b[m++] = i;
17             c[i] = 1, f[i] = 2;
18             d[i] = 1, g[i] = i + 1;
19         }
20         for (int j = 0; j < m && b[j] * i <= n; j++) {
21             int k = b[j];
22             a[i * k] = 1;
23             if (i % k == 0) {
24                 c[i * k] = c[i] + 1;
25                 f[i * k] = f[i] / c[i * k] * (c[i * k] + 1);
26                 d[i * k] = d[i];
```

```
27                g[i * k] = g[i] * k + d[i];
28                break;
29            }
30            else {
31                c[i * k] = 1;
32                f[i * k] = 2 * f[i];
33                d[i * k] = g[i];
34                g[i * k] = g[i] * (k + 1);
35            }
36        }
37    }
38 }
39
40 int main()
41 {
42    init();
43
44    int x;
45    cin >> x;
46    cout << f[x] << ' ' << g[x] << endl;
47    return 0;
48 }
```

假设输入的 x 是不超过 1000 的自然数，完成下面的判断题和单选题：

- 判断题

28. 若输入不为“1”，把第 13 行删去**不会**影响输出的结果。（ ）

29. **（2 分）**第 25 行的“f[i] / c[i * k]”可能存在无法整除而向下取整的情况。（ ）

30. **（2 分）**在执行完 init()后，f 数组不是单调递增的，但 g 数组是单调递增的。（ ）

- 单选题

31. init 函数的时间复杂度为（ ）。

A. $\Theta(n)$　　B. $\Theta(n \log n)$　　C. $\Theta(n\sqrt{n})$　　D. $\Theta(n^2)$

32. 在执行完 init()后，f[1], f[2], f[3] …… f[100]中有（ ）个等于 2。

A. 23　　B. 24　　C. 25　　D. 26

33. **（4 分）**当输入为“1000”时，输出为（ ）。

A. “15 1340”　　B. “15 2340”　　C. “16 2340”　　D. “16 1340”

三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）（Josephus 问题）有 n 个人围成一个圈，依次标号 0 至 $n-1$。从 0 号开始，依次 $0,1,0,1,\ldots$ 交替报数，报到 1 的人会离开，直至圈中只剩下一个人。求最后剩下人的编号。

试补全模拟程序。

```
01 #include <iostream>
02
03 using namespace std;
04
05 const int MAXN = 1000000;
06 int F[MAXN];
07
08 int main() {
09     int n;
10     cin >> n;
11     int i = 0, p = 0, c = 0;
12     while (①) {
13         if (F[i] == 0) {
14             if (②) {
15                 F[i] = 1;
16                 ③;
17             }
18             ④;
19         }
20         ⑤;
21     }
22     int ans = -1;
23     for (i = 0; i < n; i++)
24         if (F[i] == 0)
25             ans = i;
26     cout << ans << endl;
27     return 0;
28 }
```

34. ①处应填（ ）

A. `i < n`　　B. `c < n`　　C. `i < n - 1`　　D. `c < n - 1`

35. ②处应填（ ）

A. `i % 2 == 0`　　B. `i % 2 == 1`　　C. `p`　　D. `!p`

36. ③处应填（　）

A. `i++`　　B. `i = (i + 1) % n`

C. `c++`　　D. `p ^= 1`

37. ④处应填（　）

A. `i++`　　B. `i = (i + 1) % n`

C. `c++`　　D. `p ^= 1`

38. ⑤处应填（　）

A. `i++`　　B. `i = (i + 1) % n`

C. `c++`　　D. `p ^= 1`

（2）（矩形计数）平面上有 n 个关键点，求有多少个四条边都和 x 轴或者 y 轴平行的矩形，满足四个顶点都是关键点。给出的关键点可能有重复，但完全重合的矩形只计一次。

试补全枚举算法。

```
01 #include <iostream>
02
03 using namespace std;
04
05 struct point {
06     int x, y, id;
07 };
08
09 bool equals(point a, point b) {
10     return a.x == b.x && a.y == b.y;
11 }
12
13 bool cmp(point a, point b) {
14     return ①;
15 }
16
17 void sort(point A[], int n) {
18     for (int i = 0; i < n; i++)
19         for (int j = 1; j < n; j++)
20             if (cmp(A[j], A[j - 1])) {
21                 point t = A[j];
22                 A[j] = A[j - 1];
23                 A[j - 1] = t;
24             }
25 }
26
```

```
int unique(point A[], int n) {
    int t = 0;
    for (int i = 0; i < n; i++)
        if (②)
            A[t++] = A[i];
    return t;
}

bool binary_search(point A[], int n, int x, int y) {
    point p;
    p.x = x;
    p.y = y;
    p.id = n;
    int a = 0, b = n - 1;
    while (a < b) {
        int mid = ③;
        if (④)
            a = mid + 1;
        else
            b = mid;
    }
    return equals(A[a], p);
}

const int MAXN = 1000;
point A[MAXN];

int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> A[i].x >> A[i].y;
        A[i].id = i;
    }
    sort(A, n);
    n = unique(A, n);
    int ans = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (⑤ && binary_search(A, n, A[i].x, A[j].y) &&
                    binary_search(A, n, A[j].x, A[i].y)) {
                ans++;
            }
    cout << ans << endl;
```

```
70     return 0;
71 }
```

39. ①处应填（ ）

A. `a.x != b.x ? a.x < b.x : a.id < b.id`

B. `a.x != b.x ? a.x < b.x : a.y < b.y`

C. `equals(a, b) ? a.id < b.id : a.x < b.x`

D. `equals(a, b) ? a.id < b.id : (a.x != b.x ? a.x < b.x : a.y < b.y)`

40. ②处应填（ ）

A. `i == 0 || cmp(A[i], A[i - 1])`

B. `t == 0 || equals(A[i], A[t - 1])`

C. `i == 0 || !cmp(A[i], A[i - 1])`

D. `t == 0 || !equals(A[i], A[t - 1])`

41. ③处应填（ ）

A. `b - (b - a) / 2 + 1`

B. `(a + b + 1) >> 1`

C. `(a + b) >> 1`

D. `a + (b - a + 1) / 2`

42. ④处应填（ ）

A. `!cmp(A[mid], p)`

B. `cmp(A[mid], p)`

C. `cmp(p, A[mid])`

D. `!cmp(p, A[mid])`

43. ⑤处应填（ ）

A. `A[i].x == A[j].x`

B. `A[i].id < A[j].id`

C. `A[i].x == A[j].x && A[i].id < A[j].id`

D. `A[i].x < A[j].x && A[i].y < A[j].y`

一、单项选择题（共 15 题，每题 2 分，共计 30 分）

1	2	3	4	5	6	7	8	9	10
D	B	A	C	D	D	C	A	B	B
11	12	13	14	15					
B	A	C	B	B					

二、阅读程序（除特殊说明外，判断题 1.5 分，单选题 3 分，共计 40 分）

第 1 题	判断题（填√或×）					单选题
	16)	17)	18)	19)	20)	21)
	×	×	×	√	×	B

第 2 题	判断题（填√或×）			单选题		
	22)	23)	24)	25)	26)	27) (3.5 分)
	×	√	√	B	B	C

第 3 题	判断题（填√或×）			单选题		
	28)	29) (2 分)	30) (2 分)	31)	32)	33)(4 分)
	√	×	×	A	C	C

三、完善程序（单选题，每小题 3 分，共计 30 分）

第 1 题					第 2 题				
34)	35)	36)	37)	38)	39)	40)	41)	42)	43)
D	C	C	D	B	B	D	C	B	D