2020 CCF 非专业级别软件能力认证第一轮

（CSP-S）提高级 Pascal 语言试题

认证时间：2020 年 10 月 11 日 09:30~11:30

考生注意事项：

- 试题纸共有 14 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．请选出以下最**大**的数（　　）

A. $(550)_{10}$　　B. $(777)_{8}$　　C. 2^{10}　　D. $(22F)_{16}$

2．操作系统的功能是（　　）。

A. 负责外设与主机之间的信息交换

B. 控制和管理计算机系统的各种硬件和软件资源的使用

C. 负责诊断机器的故障

D. 将源程序编译成目标程序

3．现有一段 8 分钟的视频文件，它的播放速度是每秒 24 帧图像，每帧图像是一幅分辨率为 2048×1024 像素的 32 位真彩色图像。请问要存储这段原始无压缩视频，需要多大的存储空间？（　　）。

A. 30G　　B. 90G　　C. 150G　　D. 450G

4．今有一空栈 S，对下列待进栈的数据元素序列 a,b,c,d,e,f 依次进行：进栈，进栈，出栈，进栈，进栈，出栈的操作，则此操作完成后，栈底元素为（　　）。

A. b　　B. a　　C. d　　D. c

5．将（2, 7, 10, 18）分别存储到某个地址区间为 0~10 的哈希表中，如果哈希函数 $h(x)$ =（　　），将**不会**产生冲突，其中 a mod b 表示 a 除以 b 的余数。

A. x^2 mod 11

B. 2x mod 11

C. x mod 11

D. ⌊x/2⌋ mod 11，其中⌊x/2⌋表示 x/2 下取整

6．下列哪些问题**不能**用贪心法精确求解？（　　）

A. 霍夫曼编码问题　　B. 0-1 背包问题
C. 最小生成树问题　　D. 单源最短路径问题

7. 具有 n 个顶点，e 条边的图采用邻接表存储结构，进行深度优先遍历运算的时间复杂度为（　　）。
A. $O(n+e)$　　B. $O(n^2)$　　C. $O(e^2)$　　D. $O(n)$

8. 二分图是指能将顶点划分成两个部分，每一部分内的顶点间没有边相连的简单无向图。那么，24 个顶点的二分图**至多**有（　　）条边。
A. 144　　B. 100　　C. 48　　D. 122

9. 广度优先搜索时，一定需要用到的数据结构是（　　）。
A. 栈　　B. 二叉树　　C. 队列　　D. 哈希表

10. 一个班学生分组做游戏，如果每组三人就多两人，每组五人就多三人，每组七人就多四人，问这个班的学生人数 n 在以下哪个区间？已知 n<60。（　　）。
A. 30<n<40　　B. 40<n<50　　C. 50<n<60　　D. 20<n<30

11. 小明想通过走楼梯来锻炼身体，假设从第 1 层走到第 2 层消耗 10 卡热量，接着从第 2 层走到第 3 层消耗 20 卡热量，再从第 3 层走到第 4 层消耗 30 卡热量，依此类推，从第 k 层走到第 k+1 层消耗 10k 卡热量(k>1)。如果小明想从 1 层开始，通过连续向上爬楼梯消耗 1000 卡热量，**至少**要爬到第几层楼？（　　）。
A. 14　　B. 16　　C. 15　　D. 13

12. 表达式 a*(b+c)-d 的后缀表达形式为（　　）。
A. abc*+d-　　B. -+*abcd　　C. abcd*+-　　D. abc+*d-

13. 从一个 4×4 的棋盘中选取不在同一行也不在同一列上的两个方格，共有（　　）种方法。
A. 60　　B. 72　　C. 86　　D. 64

14. 对一个 n 个顶点、m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路时，如果不使用堆或其它优先队列进行优化，则其时间复杂度为（　　）。
A. $\Theta((m + n^2) \log n)$　　B. $\Theta(mn + n^3)$
C. $\Theta((m + n) \log n)$　　D. $\Theta(n^2)$

15. 1948 年，（　　）将热力学中的熵引入信息通信领域，标志着信息论研究的开端。
A. 欧拉（Leonhard Euler）　　B. 冯·诺伊曼（John von Neumann）
C. 克劳德·香农（Claude Shannon）　　D. 图灵（Alan Turing）

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 uses math;
02
03 var
04   n, i, j, ans: longint;
05   d: array [0..999] of longint;
06
07 begin
08   read(n);
09   for i := 0 to n - 1 do
10     read(d[i]);
11   ans := -1;
12   for i := 0 to n - 1 do
13     for j := 0 to n - 1 do
14       if d[i] < d[j] then
15         ans := max(ans, d[i] + d[j] - (d[i] and d[j]));
16   write(ans);
17 end.
```

假设输入的 n 和 d[i]都是不超过 10000 的正整数，完成下面的判断题和单选题：

- 判断题

1）n **必须小于** 1000，否则程序**可能**会发生运行错误。（　）

2）输出**一定**大于等于 0。（　）

3）若将第 13 行的“j := 0”改为“j := i + 1”，程序输出**可能**会改变。（　）

4）将第 14 行的“d[i] < d[j]”改为“d[i] <> d[j]”，程序输出**不会**改变。（　）

- 单选题

5）若输入 n 为 100，且输出为 127，则输入的 d[i]中不可能有（　）。

A. 127　　B. 126　　C. 128　　D. 125

6）若输出的数大于 0，则下面说法**正确**的是（　）。

A. 若输出为偶数，则输入的 d[i]中**最多**有两个偶数

B. 若输出为奇数，则输入的 d[i]中**至少**有两个奇数

C. 若输出为偶数，则输入的 d[i]中**至少**有两个偶数

D. 若输出为奇数，则输入的 d[i]中**最多**有两个奇数

2.

```
01 var
02   n, i, k: longint;
03   d: array[0..9999] of longint;
04
05 procedure swap(var x, y: longint);
06   var
07     z: longint;
08   begin
09     z := x;
10     x := y;
11     y := z;
12   end;
13
14 function find(L, R, k: longint): longint;
15   var
16     x, a, b: longint;
17   begin
18     x := random(R - L + 1) + L;
19     swap(d[L], d[x]);
20     a := L + 1; b := R;
21     while a < b do
22       begin
23         while (a < b) and (d[a] < d[L]) do
24           inc(a);
25         while (a < b) and (d[b] >= d[L]) do
26           dec(b);
27         swap(d[a], d[b]);
28       end;
29     if d[a] < d[L] then
30       inc(a);
31     if a - L = k then
32       exit(d[L]);
33     if a - L < k then
34       exit(find(a, R, k - (a - L)));
35     exit(find(L + 1, a - 1, k));
36   end;
37
38 begin
39   randomize;
```

```
40   read(n);
41   read(k);
42   for i := 0 to n - 1 do
43     read(d[i]);
44   write(find(0, n - 1, k));
45 end.
```

假设输入的 n，k 和 d[i]都是不超过 10000 的正整数，且 k 不超过 n，并假设 random()函数产生的是均匀的随机数，完成下面的判断题和单选题：

- 判断题

1) 第 18 行的“x”的数值范围是 L+1 到 R，即[L+1, R]。（ ）

2) 将第 29 行的“d[a]”改为“d[b]”，程序**不会**发生运行错误。（ ）

- 单选题

3) **（2.5 分）**当输入的 d[i]是严格**单调递增**序列时，第 27 行的“swap”平均执行次数是（ ）。

A. θ(n log n)　B. θ(n)　C. θ(log n)　D. θ(n^2)

4) **（2.5 分）**当输入的 d[i]是严格**单调递减**序列时，第 27 行的“swap”平均执行次数是（ ）。

A. θ(n^2)　B. θ(n)　C. θ(n log n)　D. θ(log n)

5) **（2.5 分）**若输入的 d[i]为 i，此程序①平均的时间复杂度和②最坏情况下的时间复杂度分别是（ ）。

A. θ(n), θ(n^2)　B. θ(n), θ(n log n)
C. θ(n log n), θ(n^2)　D. θ(n log n), θ(n log n)

6) **（2.5 分）**若输入的 d[i]都为同一个数，此程序平均的时间复杂度是（ ）。

A. θ(n)　B. θ(log n)　C. θ(n log n)　D. θ(n^2)

3.

```
001 const
002   maxl = 2000000000;
003 type
004   item = record
005     key: string;
006     value: longint;
007   end;
008   Map = record
009     d: array[0..maxl-1] of item;
010     cnt: longint;
```

```
011   end;
012   Queue = record
013     q: array[0..maxl-1] of string;
014     head, tail: longint;
015   end;
016 var
017   s: array[0..1] of Map;
018   q: array[0..1] of Queue;
019   st0, st1, st: string;
020   m, len, p, step: longint;
021
022 function find(var M: Map; x: string): longint;
023   var
024     i: longint;
025   begin
026     for i := 0 to M.cnt - 1 do
027       if M.d[i].key = x then
028         exit(M.d[i].value);
029     exit(-1);
030   end;
031
032 function endmark: longint;
033   begin
034     exit(-1);
035   end;
036
037 procedure insert(var M: Map; k: string; v: longint);
038   begin
039     M.d[M.cnt].key := k;
040     M.d[M.cnt].value := v;
041     inc(M.cnt);
042   end;
043
044 procedure pop(var Q: Queue);
045   begin
046     inc(Q.head);
047   end;
048
049 function front(var Q: Queue): string;
050   begin
051     exit(Q.q[Q.head + 1]);
052   end;
053
```

```
function empty(var Q: Queue): boolean;
  begin
    exit(Q.head = Q.tail);
  end;

procedure push(var Q: Queue; x: string);
  begin
    inc(Q.tail);
    Q.q[Q.tail] := x;
  end;

function LtoR(s: string; L, R: longint): string;
  var
    t: string;
    tmp: char;
    i: longint;
  begin
    inc(L); inc(R);
    t := s;
    tmp := t[L];
    for i := L to R - 1 do
      t[i] := t[i + 1];
    t[R] := tmp;
    exit(t);
  end;

function RtoL(s: string; L, R: longint): string;
  var
    t: string;
    tmp: char;
    i: longint;
  begin
    inc(L); inc(R);
    t := s;
    tmp := t[R];
    for i := R downto L + 1 do
      t[i] := t[i - 1];
    t[L] := tmp;
    exit(t);
  end;

function check(st: string; p, step: longint): boolean;
  begin
```

```
097     if find(s[p], st) <> endmark then exit(false);
098     inc(step);
099     if find(s[p xor 1], st) = endmark then
100       begin
101         insert(s[p], st, step);
102         push(q[p], st);
103         exit(false);
104       end;
105     writeln(find(s[p xor 1], st) + step);
106     exit(true);
107   end;
108
109 begin
110   readln(st0); readln(st1);
111   len := length(st0);
112   if len <> length(st1) then
113     begin
114       writeln(-1);
115       exit;
116     end;
117   if st0 = st1 then
118     begin
119       writeln(0);
120       exit;
121     end;
122   read(m);
123   insert(s[0], st0, 0); insert(s[1], st1, 0);
124   push(q[0], st0); push(q[1], st1);
125   p := 0;
126   while not (empty(q[0]) and empty(q[1])) do
127     begin
128       st := front(q[p]); pop(q[p]);
129       step := find(s[p], st);
130       if (((p = 0) and
131           (check(LtoR(st, m, len - 1), p, step) or
132            check(RtoL(st, 0, m), p, step)))
133             or
134           ((p = 1) and
135           (check(LtoR(st, 0, m), p, step) or
136            check(RtoL(st, m, len - 1), p, step)))) then
137         exit;
138       p := p xor 1;
139     end;
```

```
140   writeln(-1);
141 end.
```

- 判断题

1) 输出**可能**为 0。（ ）

2) 若输入的两个字符串长度均为 101 时，则 m=0 时的输出与 m=100 时的输出是一样的。（ ）

3) 若两个字符串的长度均为 n，则最坏情况下，此程序的时间复杂度为 θ(n!)。（ ）

- 单选题

4) **（2.5 分）**若输入的第一个字符串长度由 100 个不同的字符构成，第二个字符串是第一个字符串的倒序，输入的 m 为 0，则输出为（ ）。

A. 49　　B. 50　　C. 100　　D. -1

5) **（4 分）**已知当输入为“0123\n3210\n1”时输出为 4，当输入为“012345\n543210\n1”时输出为 14，当输入为“01234567\n76543210\n1”时输出为 28，则当输入为“0123456789ab\nba9876543210\n1”输出为（ ）。其中“\n”为换行符。

A. 56　　B. 84　　C. 102　　D. 68

6) **（4 分）**若两个字符串的长度均为 n，且 0<m<n-1，且两个字符串的构成相同（即任何一个字符在两个字符串中出现的次数均相同），则下列说法**正确**的是（ ）。提示：考虑输入与输出有多少对字符前后顺序不一样。

A. 若 n、m 均为奇数，则输出**可能**小于 0。
B. 若 n、m 均为偶数，则输出**可能**小于 0。
C. 若 n 为奇数、m 为偶数，则输出**可能**小于 0。
D. 若 n 为偶数、m 为奇数，则输出**可能**小于 0。

三、完善程序（单选题，每小题 3 分，共计 30 分）

1. **（分数背包）**小 S 有 n 块蛋糕，编号从 1 到 n。第 i 块蛋糕的价值是w_i，体积是v_i。他有一个大小为 B 的盒子来装这些蛋糕，也就是说装入盒子的蛋糕的体积总和不能超过 B。

他打算选择一些蛋糕装入盒子，他希望盒子里装的蛋糕的价值之和尽量大。

为了使盒子里的蛋糕价值之和更大，他可以任意切割蛋糕。具体来说，他可以选择一个 α（0<α<1），并将一块价值是 w，体积为 v 的蛋糕切割成两

块，其中一块的价值是$\alpha \cdot w$，体积是$\alpha \cdot v$，另一块的价值是$(1-\alpha) \cdot w$，体积是$(1-\alpha) \cdot v$。他可以重复无限次切割操作。
现要求编程输出最大可能的价值，以分数的形式输出。
比如 n=3，B=8，三块蛋糕的价值分别是 4、4、2，体积分别是 5、3、2。那么最优的方案就是将体积为 5 的蛋糕切成两份，一份体积是 3，价值是 2.4，另一份体积是 2，价值是 1.6，然后把体积是 3 的那部分和后两块蛋糕打包进盒子。最优的价值之和是 8.4，故程序输出 42/5。
输入的数据范围为： $1 \leqslant n \leqslant 1000$， $1 \leqslant B \leqslant 10^5$；$1 \leqslant w_i, v_i \leqslant 100$。
提示：将所有的蛋糕按照性价比w_i/v_i从大到小排序后进行贪心选择。
试补全程序。

```
01 const
02   maxn = 1005;
03
04  var
05   n, B, i, j, curV, curW: longint;
06   w, v: array[0..maxn-1] of longint;
07
08 function gcd(u, v: longint): longint;
09   begin
10     if v = 0 then
11       exit(u);
12     exit(gcd(v, u mod v));
13   end;
14
15 procedure print(w, v: longint);
16   var
17     d: longint;
18   begin
19     d := gcd(w, v);
20     w := w div d;
21     v := v div d;
22     if (v = 1) then
23       writeln(w)
24     else
25       writeln(w, '/', v);
26   end;
27
28 procedure swap(var x, y: longint);
29   var
30     t: longint;
31   begin
32     t := x; x := y; y := t;
```

```
33   end;
34
35 begin
36   read(n, B);
37   for i := 1 to n do
38     read(w[i], v[i]);
39   for i := 1 to n - 1 do
40     for j := 1 to n - 1 do
41       if ① then
42         begin
43           swap(w[j], w[j + 1]);
44           swap(v[j], v[j + 1]);
45         end;
46   if ② then
47     begin
48       ③
49     end
50   else
51     begin
52       print(B * w[1], v[1]);
53       exit;
54     end;
55
56   for i:= 2 to n do
57     if curV + v[i] <= B then
58       begin
59         curV := curV + v[i];
60         curW := curW + w[i];
61       end
62     else
63       begin
64         print(④);
65         exit;
66       end;
67   print(⑤);
68 end.
```

1) ①处应填（　）

A. `w[j] / v[j] < w[j + 1] / v[j + 1]`
B. `w[j] / v[j] > w[j + 1] / v[j + 1]`
C. `v[j] * w[j + 1] < v[j + 1] * w[j]`
D. `w[j] * v[j + 1] < w[j + 1] * v[j]`

2) ②处应填（　）

A. `w[1] <= B`　　B. `v[1] <= B`　　C. `w[1] >= B`　　D. `v[1] >= B`

3) ③处应填（　）

A.
```
print(v[1], w[1]); exit;
```
B.
```
curV := 0; curW := 0;
```
C.
```
print(w[1], v[1]); exit;
```
D.
```
curV := v[1]; curW := w[1];
```

4) ④处应填（　）

A.
```
curW * v[i] + curV * w[i], v[i]
```
B.
```
(curW - w[i]) * v[i] + (B - curV) * w[i], v[i]
```
C.
```
curW + v[i], w[i]
```
D.
```
curW * v[i] + (B - curV) * w[i], v[i]
```

5) ⑤处应填（　）

A. `curW, curV`　　B. `curW, 1`

C. `curV, curW`　　D. `curV, 1`

2. （**最优子序列**）取 $m = 16$，给出长度为n的整数序列$a_1, a_2, \cdots, a_n (0 \le a_i < 2^m)$。对于一个二进制数x，定义其分值w(x)为$x + popcnt(x)$，其中popcnt(x)表示 x 二进制表示中 1 的个数。对于一个子序列$b_1, b_2, \cdots, b_k$，定义其子序列分值S为$w(b_1 \oplus b_2) + w(b_2 \oplus b_3) + w(b_3 \oplus b_4) + \cdots + w(b_{k-1} \oplus b_k)$。其中$\oplus$表示按位异或。对于空子序列，规定其子序列分值为0。求一个子序列使得其子序列分值最大，输出这个最大值。

输入第一行包含一个整数 $n (1 \le n \le 40000)$。接下来一行包含n个整数$a_1, a_2, \cdots, a_n$。

提示：考虑优化朴素的动态规划算法，将前$\frac{m}{2}$位和后$\frac{m}{2}$位分开计算。

`Max[x][y]` 表示当前的子序列下一个位置的高 8 位是 x、最后一个位置的低 8 位是 y 时的最大价值。

试补全程序。

```
01 const
02   MAXN = 40000; M = 16; B = M shr 1; MS = (1 shl B) - 1;
03   INF = 1000000000000000;
04
05 var
06   Max: array[0..MS+3, 0..MS+3] of int64;
07   n, x, y, z, i: longint;
08   ans, a, v: int64;
```

```
09
10 function w(x: longint): longint;
11   var
12     s: longint;
13   begin
14     s := x;
15     while x <> 0 do
16       begin
17         ①;
18         inc(s);
19       end;
20     exit(s);
21   end;
22
23 procedure to_max(var x: int64; y: int64);
24   begin
25     if x < y then
26       x := y;
27   end;
28
29 begin
30   ans := 0;
31   read(n);
32   for x := 0 to MS do
33     for y := 0 to MS do
34       Max[x][y] := -INF;
35   for i := 1 to n do
36     begin
37       read(a);
38.      x := ②; y := a and MS;
39       v := ③;
40       for z := 0 to MS do
41         to_max(v, ④);
42       for z := 0 to MS do
43         ⑤;
44       to_max(ans, v);
45     end;
46   writeln(ans);
47 end.
```

1) ①处应填（　）

A. `x := x shr 1`

B. `x := x xor (x and (x xor (x + 1)))`

C. `x := x - (x or -x)`

D. `x := x xor (x and (x xor (x - 1)))`

2) ②处应填（ ）

A. `(a and MS) shl B`

B. `a shr B`

C. `a and (1 shl B)`

D. `a and (MS shl B)`

3) ③处应填（ ）

A. `-INF`

B. `Max[y][x]`

C. `0`

D. `Max[x][y]`

4) ④处应填（ ）

A. `Max[x][z] + w(y xor z)`

B. `Max[x][z] + w(a xor z)`

C. `Max[x][z] + w(x xor (z shl B))`

D. `Max[x][z] + w(x xor z)`

5) ⑤处应填（ ）

A. `to_max(Max[y][z], v + w(a xor (z shl B)))`

B. `to_max(Max[z][y], v + w((x xor z) shl B))`

C. `to_max(Max[z][y], v + w(a xor (z shl B)))`

D. `to_max(Max[x][z], v + w(y xor z))`