2021 CCF 非专业级别软件能力认证第一轮
（CSP-S1）提高级 Pascal 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 17 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。

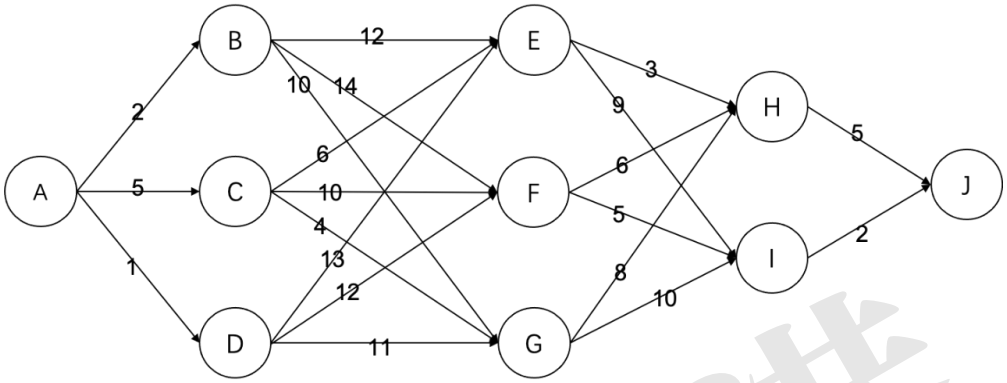
A. 81

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 uses math;
02
```

```
03 var
04     a1, b1, c1, d1: longint;
05     a2, b2, c2, d2: longint;
06     t: longint;
07     r, x, y: double;
08
09 function sq(const x: longint): longint;
10     begin
11         sq := x * x;
12     end;
13
14 function cu(const x: longint): longint;
15     begin
16         cu := x * x * x;
17     end;
18
19 begin
20     r := arccos(0.5);
21
22     read(a1, b1, c1, d1);
23     read(a2, b2, c2, d2);
24
25     t := sq(a1 - a2) + sq(b1 - b2) + sq(c1 - c2);
26
27     if (t <= sq(d2 - d1)) then
28         write(cu(min(d1, d2)) * r * 4 : 0 : 4)
29     else if (t >= sq(d2 + d1)) then
30         write(0)
31     else begin
32         x := d1 - (sq(d1) - sq(d2) + t) / sqrt(t) / 2;
33         y := d2 - (sq(d2) - sq(d1) + t) / sqrt(t) / 2;
34         write((x * x * (3 * d1 - x) + y * y * (3 * d2 - y)) *
                                                r : 0 : 4);
35     end;
36     writeln;
37 end.
```

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 06 行中 t 的类型声明从 longint 改为 double，**不会**影响程序运行的结果。（ ）

17. 将第 32、33 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 34 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 uses math;
02 type
03     Node = record
04         h, j, m, w: longint;
05     end;
06
07 var
08     n, i: longint;
09     a: array[0..1004] of longint;
10
11 function _Node(const _h, _j, _m, _w: longint): Node;
12     begin
13         _Node.h := _h;
14         _Node.j := _j;
15         _Node.m := _m;
16         _Node.w := _w;
17     end;
18
19 function add(const a,b: Node): Node;
20     begin
21         add := _Node(
22             max(a.h, a.w + b.h),
23             max(max(a.j, b.j), a.m + b.h),
24             max(a.m + b.w, b.m),
25             a.w + b.w
26         );
27     end;
28
29 function solve1(h, m: longint): Node;
30     var
31         j: longint;
32     begin
33         if (h > m) then
```

```
34              exit(_Node(-1, -1, -1, -1));
35          if (h = m) then
36              exit(_Node(max(a[h], 0), max(a[h], 0), max(a[h], 0),
                                                          a[h]));
37          j := (h + m) shr 1;
38          solve1 := add(solve1(h, j), solve1(j + 1, m));
39      end;
40
41  function solve2(h, m: longint): longint;
42      var
43          i, j, wh, wm, wht, wmt: longint;
44      begin
45          if (h > m) then
46              exit(-1);
47          if (h = m) then
48              exit(max(a[h], 0));
49          j := (h + m) shr 1;
50          wh := 0; wm := 0;
51          wht := 0; wmt := 0;
52          for i := j downto h do
53              begin
54                  wht := wht + a[i];
55                  wh := max(wh, wht);
56              end;
57          for i := j + 1 to m do
58              begin
59                  wmt := wmt + a[i];
60                  wm := max(wm, wmt);
61              end;
62          solve2 := max(max(solve2(h, j), solve2(j + 1, m)), wh +
                                                              wm);
63      end;
64
65  begin
66      read(n);
67      for i := 1 to n do read(a[i]);
68      writeln(solve1(1, n).j);
69      writeln(solve2(1, n));
70  end.
```

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 34 行与第 46 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 var
02    base: array[0..63] of shortint;
03    table: array[0..255] of shortint;
04    i, opt, len: longint;
05    str, ret: string;
06
07 procedure init;
08     begin
09         for i := 0 to 25 do base[i] := ord('A') + i;
10         for i := 0 to 25 do base[26 + i] := ord('a') + i;
11         for i := 0 to 9 do base[52 + i] := ord('0') + i;
12         base[62] := ord('+'); base[63] := ord('/');
13
14         for i := 0 to 255 do table[i] := $ff;
15         for i := 0 to 63 do table[base[i]] := i;
16         table[ord('=')] := 0;
17     end;
18
19 function encode(str: string): string;
20     begin
21         ret := '';
22         i := 1;
23         len := length(str);
24         while (i + 3 <= len + 1) do
25             begin
26                 ret := ret + chr(base[ord(str[i]) shr 2]);
27                 ret := ret + chr(base[(ord(str[i]) and $03) shl 4
                                      or ord(str[i + 1]) shr 4]);
28                 ret := ret + chr(base[(ord(str[i + 1]) and $0f) shl
                                         2 or ord(str[i + 2]) shr 6]);
29                 ret := ret + chr(base[ord(str[i + 2]) and $3f]);
30                 i := i + 3;
31             end;
32         if (i <= len) then
```

```
33              begin
34                  ret := ret + chr(base[ord(str[i]) shr 2]);
35                  if (i = len) then
36                      begin
37                          ret := ret + chr(base[(ord(str[i]) and
                                                   $03) shl 4]);
38                          ret := ret + '==';
39                      end
40                  else
41                      begin
42                          ret := ret + chr(base[(ord(str[i]) and $03)
                                    shl 4 or ord(str[i + 1]) shr 4]);
43                          ret := ret + chr(base[(ord(str[i + 1])
                                                 and $0f) shl 2]);
44                          ret := ret + '=';
45                      end;
46              end;
47          encode := ret;
48      end;
49
50  function decode(str: string): string;
51      begin
52          ret := '';
53          len := length(str);
54          i := 1;
55          while (i <= len) do
56              begin
57                  ret := ret + chr(table[ord(str[i])] shl 2 or
                                        table[ord(str[i + 1])] shr 4);
58                  if (str[i + 2] <> '=') then
59                      ret := ret + chr((table[ord(str[i + 1])] and
                          $0f) shl 4 or table[ord(str[i + 2])] shr 2);
60                  if (str[i + 3] <> '=') then
61                      ret := ret + chr(table[ord(str[i + 2])] shl
                                           6 or table[ord(str[i + 3])]);
62                  i := i + 4;
63              end;
64          decode := ret;
65      end;
66
67  begin
68      init;
69      writeln(longint(table[0]));
70
71      read(opt, str);
72      str := copy(str, 2, length(str) - 1);
73
74      if (opt <> 0) then
75          writeln(decode(str))
```

```
76      else
77          writeln(encode(str));
78  end.
```

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 uses math;
02 const
03     M = 10000;
04 var
05     Vis: array[0..M] of boolean;
06     F: array[0..M] of longint;
07     n, i, r, x, t: longint;
```

```
08
09 procedure update(var x: longint; y: longint);
10     begin
11         if (y < x) then x := y;
12     end;
13
14 begin
15     read(n);
16     for i := 0 to M do
17         F[i] := MaxLongint;
18     ①;
19     r := 0;
20     while (②) do
21         begin
22             inc(r);
23             x := 0;
24             for i := 1 to M do
25                 if (③) then
26                     x := i;
27             Vis[x] := true;
28             for i := 1 to M do
29                 if (④) then
30                     begin
31                         t := F[i] + F[x];
32                         if (i + x <= M) then
33                             update(F[i + x], t);
34                         if (i <> x) then
35                             update(F[abs(i - x)], t);
36                         if (i mod x = 0) then
37                             update(F[i div x], t);
38                         if (x mod i = 0) then
39                             update(F[x div i], t);
40                     end;
41         end;
42     writeln(F[n]);
43 end.
```

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. `not Vis[n]`　　B. `r < n`

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

36. ③处应填（　）

A. `F[i] = r`　　B. `not Vis[i] and (F[i] = r)`

C. `F[i] < F[x]`　　D. `not Vis[i] and (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 $a_0,\ldots,a_{n-1}$，和 m 次询问，每次询问给定 l,r，求 $\max\{a_l,\ldots,a_r\}$。

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 uses math;
002 type
003     pointer = ^node;
004     node = record
005         val: longint;
006         dep, dfn, right: longint;
007         son: array[0..1] of pointer; // son[0], son[1] 分别表示
                                                          左右儿子
008     end;
009 const
010     MAXN = 100000;
011     MAXT = MAXN shl 1;
012     MAXL = 18;
```

```
013     MAXB = 9;
014     MAXC = MAXT div MAXB;
015 var
016     T: array[0..MAXN-1] of node;
017     n, _t, b, c, i, m, l, r: longint;
018     _Log2, Dif: array[0..MAXC] of longint;
019     Pos: array[0..(1 shr (MAXB - 1))+4] of longint;
020     root: pointer;
021     A: array[0..MAXT-1] of pointer;
022     Min: array[0..MAXL-1, 0..MAXC-1] of pointer;
023     S: array[0..MAXN] of pointer;
024
025 procedure build; // 建立 Cartesian 树
026     var
027         top: longint = 0;
028         i: longint;
029         p: pointer;
030     begin
031         for i := 0 to n - 1 do
032             begin
033                 p := @T[i];
034                 while (top <> 0) and (S[top]^.val < p^.val) do
035                     begin
036                         ①
037                     end;
038                 if (top <> 0) then
039                     ②;
040                 inc(top);
041                 S[top] := p;
042             end;
043         root := S[1];
044     end;
045
046 procedure DFS(p: pointer); // 构建 Euler 序列
047     var
048         i: longint;
049     begin
050         p^.dfn := _t;
051         inc(_t);
052         A[p^.dfn] := p;
053         for i := 0 to 1 do
054             begin
055                 if (p^.son[i] <> nil) then
056                     begin
057                         p^.son[i]^.dep := p^.dep + 1;
058                         DFS(p^.son[i]);
059                         A[_t] := p;
060                         inc(_t);
061                     end;
```

```
062             end;
063         p^.right := _t - 1;
064     end;
065
066 function _min(x, y: pointer): pointer;
067     begin
068         if (③) then exit(x) else exit(y);
069     end;
070
071 procedure ST_init;
072     var
073         i, j, l: longint;
074     begin
075         b := ceil((log2(_t) / 2));
076         c := _t div b;
077         _Log2[1] := 0;
078         for i := 2 to c do
079             _Log2[i] := _Log2[i shr 1] + 1;
080         for i := 0 to c - 1 do
081             begin
082                 Min[0][i] := A[i * b];
083                 for j := 1 to b - 1 do
084                     Min[0][i] := _min(Min[0][i], A[i * b + j]);
085             end;
086         i := 1; l := 2;
087         while (l <= c) do
088             begin
089                 for j := 0 to c - l do
090                     Min[i][j] := _min(Min[i - 1][j], Min[i – 1]
                                                    [j + (l shr 1)]);
091                 inc(i);
092                 l := l shl 1;
093             end;
094     end;
095
096 procedure small_init; // 块内预处理
097     var
098         i, j, mx, v, range, S: longint;
099     begin
100         for i := 0 to c do
101             begin
102                 if (b - 1 < _t - i * b - 1) then range := b - 1
                                    else range := _t - i * b - 1;
103                 for j := 1 to range do
104                     if (④) then
105                         Dif[i] := Dif[i] or (1 shl (j - 1));
106             end;
107         for S := 0 to (1 shl (b - 1)) - 1 do
108             begin
```

```
109                 mx := 0; v := 0;
110                 for i := 1 to b - 1 do
111                     begin
112                         ⑤;
113                         if (v < mx) then
114                             begin
115                                 mx := v;
116                                 Pos[S] := i;
117                             end;
118                     end;
119             end;
120     end;
121
122 function ST_query(l, r: longint): pointer;
123     var
124         g: longint;
125     begin
126         g := _Log2[r - l + 1];
127         exit(_min(Min[g][l], Min[g][r - (1 shl g) + 1]));
128     end;
129
130 function small_query(l, r: longint): pointer; // 块内查询
131     var
132         p, S: longint;
133     begin
134         p := l div b;
135         S := ⑥;
136         exit(A[l + Pos[S]]);
137     end;
138
139 function query(l, r: longint): pointer;
140     var
141         pl, pr: longint;
142         s: pointer;
143     begin
144         if (l > r) then exit(query(r, l));
145         pl := l div b; pr := r div b;
146         if (pl = pr) then
147             exit(small_query(l, r))
148         else
149             begin
150                 s := _min(small_query(l, pl * b + b - 1),
                                          small_query(pr * b, r));
151                 if (pl + 1 <= pr - 1) then
152                     s := _min(s, ST_query(pl + 1, pr - 1));
153                 exit(s);
154             end;
155     end;
156
```

```
157 begin
158     read(n, m);
159     for i := 0 to n - 1 do
160         read(T[i].val);
161     build;
162     DFS(root);
163     ST_init;
164     small_init;
165     for i := 1 to m do
166         begin
167             read(l, r);
168             writeln(query(T[l].dfn, T[r].dfn)^.val);
169         end;
170 end.
```

38. ①处应填（　）

A. `p^.son[0] := S[top]; dec(top);`　　B. `p^.son[1] := S[top]; dec(top);`

C. `S[top]^.son[0] := p; dec(top);`　　D. `S[top]^.son[1] := p; dec(top);`

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 := v + 1 - (S shr i and 1) * 2`

B. `v := v + (S shr i and 1) * 2 - 1`

C. `v := v + (S shr (i - 1) and 1) * 2 - 1`

D. `v := v + 1 - (S shr (i - 1) and 1) * 2`

43. ⑥处应填（　）

A. `(Dif[p] shr (r - p * b)) and ((1 shl (r - l)) - 1)`

B. `Dif[p]`

C. `(Dif[p] shr (l - p * b)) and ((1 shl (r - l)) - 1)`

D. `(Dif[p] shr ((p + 1) * b - r)) and ((1 shl (r - l + 1)) - 1)`