2021 CCF 非专业级别软件能力认证第一轮

（CSP-S1）提高级 C++语言试题解析

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

[参考答案]A

2、二进制数 001010102和 000101102 的和为（ ）。  
A. 001111002 B. 010000002 C. 001111002 D. 010000102

[参考答案]B

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

[参考答案]A

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

[参考答案]C

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

[参考答案]C

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)=x2 mod 11。请问 7 存储在哈希表哪个地址中（ ）。  
A. 5  
B. 6  
C. 7  
D. 8

[参考答案]C

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

[参考答案]C

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

[参考答案]B

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

[参考答案]D

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

[参考答案]A

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

[参考答案]A

12、斐波那契数列的定义为：，，F1=1，F2=1，Fn=Fn−1+Fn−2(n>=3) 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()  
B. O(!)  
C. O(2n)  
D. O(log)

[参考答案]C

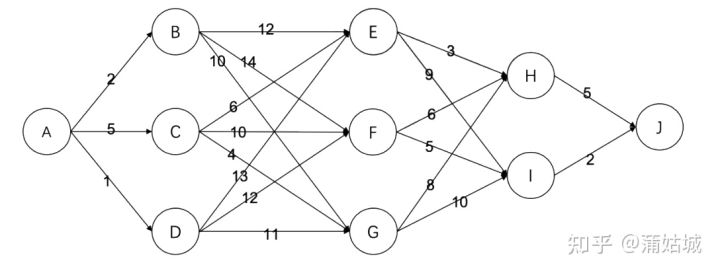
13、有 8 个苹果从左到右排成一排，你要从中挑选至少一个苹果，并且不能同时挑选相邻的两 个苹果，一共有（ ）种方案。  
A. 36 B. 48 C. 54 D. 64

[参考答案]C

14、设一个三位数 n=，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

[参考答案]C

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

[参考答案]B

## 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”

[参考答案]D

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

[参考答案]C

（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 }

**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“5−1011−95−7”时，输出的第二行为“7”。（×）

* 单选题

25、solve1(1, n) 的时间复杂度为（ ）。

A. O(log)

B. O()

C. O(log)

D. O(!)

[参考答案]B

26、solve2(1, n) 的时间复杂度为（ ）。

A. O(log)

B. O()

C. O(log)

D. O(!)

[参考答案]C

27、当输入为“10−32100−89−4−594”时，输出的第一行为（ ）。

A. “13”

B. “17”

C. “24”

D. “12”

[参考答案]B

（3）

**#include** <iostream>

**#include** <string>

**using** **namespace** std;

**char** base[64];

**char** table[256];

**void** **init**()

{

**for** (**int** i **=** 0; i **<** 26; i**++**) base[i] **=** 'A' **+** i;

**for** (**int** i **=** 0; i **<** 26; i**++**) base[26 **+** i] **=** 'a' **+** i;

**for** (**int** i **=** 0; i **<** 10; i**++**) base[52 **+** i] **=** '0' **+** i;

base[62] **=** '+', base[63] **=** '/';

**for** (**int** i **=** 0; i **<** 256; i**++**) table[i] **=** 0xff;

**for** (**int** i **=** 0; i **<** 64; i**++**) table[base[i]] **=** i;

table['='] **=** 0;

}

string **encode**(string str)

{

string ret;

**int** i;

**for** (i **=** 0; i **+** 3 **<=** str.size(); i **+=** 3) {

ret **+=** base[str[i] **>>** 2];

ret **+=** base[(str[i] **&** 0x03) **<<** 4 **|** str[i **+** 1] **>>** 4];

ret **+=** base[(str[i **+** 1] **&** 0x0f) **<<** 2 **|** str[i **+** 2] **>>** 6];

ret **+=** base[str[i **+** 2] **&** 0x3f];

}

**if** (i **<** str.size()) {

ret **+=** base[str[i] **>>** 2];

**if** (i **+** 1 **==** str.size()) {

ret **+=** base[(str[i] **&** 0x03) **<<** 4];

ret **+=** "==";

}

**else** {

ret **+=** base[(str[i] **&** 0x03) **<<** 4 **|** str[i **+** 1] **>>** 4];

ret **+=** base[(str[i **+** 1] **&** 0x0f) **<<** 2];

ret **+=** "=";

}

}

**return** ret;

}

string **decode**(string str){

string ret;

**int** i;

**for** (i **=** 0; i **<** str.size(); i **+=** 4) {

ret **+=** table[str[i]] **<<** 2 **|** table[str[i **+** 1]] **>>** 4;

**if** (str[i **+** 2] **!=** '=')

ret **+=** (table[str[i **+** 1]] **&** 0x0f) **<<** 4 **|** table[str[i **+** 2]] **>>** 2;

**if** (str[i **+** 3] **!=** '=')

ret **+=** table[str[i **+** 2]] **<<** 6 **|** table[str[i **+** 3]];

}

**return** ret;

}

**int** **main**()

{

init();

cout **<<** **int**(table[0]) **<<** endl;

**int** opt;

string str;

cin **>>** opt **>>** str;

cout **<<** (opt **?** decode(str) **:** encode(str)) **<<** endl;

**return** 0;

}

**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. O(n)

B. O(n)

C. O(nlogn)

D. O(n!)

[参考答案]B

32、输出的第一行为（ ）。

A. “0xff”

B. “255”

C. “0xFF”

D. “-1”

[参考答案]D

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

A. “Q1NQMjAyMWNzcAv=”

B. “Q1NQMjAyMGNzcA==”

C. “Q1NQMjAyMGNzcAv=”

D. “Q1NQMjAyMWNzcA==”

[参考答案]D

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

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

试补全程序。

**#include** <iostream>

**#include** <cstdlib>

**#include** <climits>

**using** **namespace** std;

**const** **int** M **=** 10000;

**bool** Vis[M **+** 1];

**int** F[M **+** 1];

**void** **update**(**int** **&**x, **int** y) {

**if** (y **<** x)

x **=** y;

}

**int** **main**() {

**int** n;

cin **>>** n;

**for** (**int** i **=** 0; i **<=** M; i**++**)

F[i] **=** INT\_MAX;

①;

**int** r **=** 0;

**while** (②) {

r**++**;

**int** x **=** 0;

**for** (**int** i **=** 1; i **<=** M; i**++**)

**if** (③)

x **=** i;

Vis[x] **=** 1;

**for** (**int** i **=** 1; i **<=** M; i**++**)

**if** (④) {

**int** t **=** F[i] **+** F[x];

**if** (i **+** x **<=** M)

update(F[i **+** x], t);

**if** (i **!=** x)

update(F[abs(i **-** x)], t);

**if** (i **%** x **==** 0)

update(F[i **/** x], t);

**if** (x **%** i **==** 0)

update(F[x **/** i], t);

}

}

cout **<<** F[n] **<<** endl;

**return** 0;

}

34、①处应填（ ）

A. F[4] = 0

B. F[1] = 4

C. F[1] = 2

D. F[4] = 1

[参考答案]D

35、②处应填（ ）

A. !Vis[n]

B. r < n

C. F[M] == INT\_MAX

D. F[n] == INT\_MAX

[参考答案]A

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]

[参考答案]D

37、④处应填（ ）

A. F[i] < F[x]

B. F[i] <= r

C. Vis[i]

D. i <= x

[参考答案]C

(2) （ RMQ 区间最值问题） 给定序列 0,…,n−1，和  次询问，每次询问给定 ，，求max{l,…,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**(+)**， 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=[log2t2]。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(tblogt)=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 2b−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(b2b)，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。

**#include** <iostream>

**#include** <cmath>

**using** **namespace** std;

**const** **int** MAXN **=** 100000,

MAXT **=** MAXN **<<** 1;

**const** **int** MAXL **=** 18, MAXB **=** 9, MAXC **=** MAXT **/** MAXB;

**struct** **node** {

**int** val;

**int** dep, dfn, end;

node **\***son[2]; *// son[0], son[1] 分别表示左右儿子*

} T[MAXN];

**int** n, t, b, c, Log2[MAXC **+** 1];

**int** Pos[(1 **<<** (MAXB **-** 1)) **+** 5], Dif[MAXC **+** 1];

node **\***root, **\***A[MAXT], **\***Min[MAXL][MAXC];

**void** **build**() { *// 建立 Cartesian 树*

**static** node **\***S[MAXN **+** 1];

**int** top **=** 0;

**for** (**int** i **=** 0; i **<** n; i**++**) {

node **\***p **=** **&**T[i];

**while** (top **&&** S[top]**->**val **<** p**->**val)

①;

**if** (top)

②;

S[**++**top] **=** p;

}

root **=** S[1];

}

**void** **DFS**(node **\***p) { *// 构建 Euler 序列*

A[p**->**dfn **=** t**++**] **=** p;

**for** (**int** i **=** 0; i **<** 2; i**++**)

**if** (p**->**son[i]) {

p**->**son[i]**->**dep **=** p**->**dep **+** 1;

DFS(p**->**son[i]);

A[t**++**] **=** p;

}

p**->**end **=** t **-** 1;}

node **\*min**(node **\***x, node **\***y) {

**return** ③ **?** x : y;

}

**void** **ST\_init**() {

b **=** (**int**)(ceil(log2(t) **/** 2));

c **=** t **/** b;

Log2[1] **=** 0;

**for** (**int** i **=** 2; i **<=** c; i**++**)

Log2[i] **=** Log2[i **>>** 1] **+** 1;

**for** (**int** i **=** 0; i **<** c; i**++**) {

Min[0][i] **=** A[i **\*** b];

**for** (**int** j **=** 1; j **<** b; j**++**)

Min[0][i] **=** min(Min[0][i], A[i **\*** b **+** j]);

}

**for** (**int** i **=** 1, l **=** 2; l **<=** c; i**++**, l **<<=** 1)

**for** (**int** j **=** 0; j **+** l **<=** c; j**++**)

Min[i][j] **=** min(Min[i **-** 1][j], Min[i **-** 1][j **+** (l **>>**1)]);

}

**void** **small\_init**() { *// 块内预处理* **for** (**int** i **=** 0; i **<=** c; i**++**)

**for** (**int** j **=** 1; j **<** b **&&** i **\*** b **+** j **<** t; j**++**)

**if** (④)

Dif[i] **|=** 1 **<<** (j **-** 1);

**for** (**int** S **=** 0; S **<** (1 **<<** (b **-** 1)); S**++**) {

**int** mx **=** 0, v **=** 0;

**for** (**int** i **=** 1; i **<** b; i**++**) {

⑤;

**if** (v **<** mx) {

mx **=** v;

Pos[S] **=** i;

}

}

}

}

node **\*ST\_query**(**int** l, **int** r) {

**int** g **=** Log2[r **-** l **+** 1];

**return** min(Min[g][l], Min[g][r **-** (1 **<<** g) **+** 1]);

}

node **\*small\_query**(**int** l, **int** r) { *// 块内查询*

**int** p **=** l **/** b;

**int** S **=** ⑥;

**return** A[l **+** Pos[S]];

}

node **\*query**(**int** l, **int** r) {

**if** (l **>** r)

**return** query(r, l);

**int** pl **=** l **/** b, pr **=** r **/** b;

**if** (pl **==** pr) {

**return** small\_query(l, r);

} **else** {

node **\***s **=** min(small\_query(l, pl **\*** b **+** b **-** 1),small\_query(pr **\*** b, r));

**if** (pl **+** 1 **<=** pr **-** 1)

s **=** min(s, ST\_query(pl **+** 1, pr **-** 1));

**return** s;

}

}

**int** **main**() {

**int** m;

cin **>>** n **>>** m;

**for** (**int** i **=** 0; i **<** n; i**++**)

cin **>>** T[i].val;

build();

DFS(root);

ST\_init();

small\_init();

**while** (m**--**) {

**int** l, r;

cin **>>** l **>>** r;

cout **<<** query(T[l].dfn, T[r].dfn)**->**val **<<** endl;

}

**return** 0;}

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

[参考答案]A

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

[参考答案]D

40、③处应填（ ）

A. x->dep < y->dep

B. x < y

C. x->dep > y->dep

D. x->val < y->val

[参考答案]A

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

[参考答案]D

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

[参考答案]D

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)

[参考答案]C

一、单项选择题

1. 答案：A

解析：ls，大纲里要求在 Linux系统终端中使用cd、pwd、ls等命令更改、显示目录路径和查看目录中的文件

2. 答案：B

解析：和的规则为1+1=10，0+1=1，0+0=0，注意进位。

3. 答案：A

解析：栈溢出，上课过程中天天强调。

4. 答案：C

解析：比赛前着重讲了，稳定性指的是对需要排序的数据元素序列，将其按关键字进行排序，若相同关键字元素之间的位置关系，排序前与排序后的相对位置不发生变化，称此排序方法满足稳定性;否则称这种排序方法满足不稳定性。堆的话只分左右孩子，当然不满足。

5. 答案：C

解析：2\*n个数找最大值，直接找需要(2\*n-1)\*2次比较，但是过程中如果他比最大值要大，最小值就不需要比较了，否则最大不用比较。

将n个数分为n对，两个谁大谁小1次，共n次。

接下来n个分别求最大最小，共(n−1)∗2(n−1)∗2次。

总次数为3n\*2

6. 答案：C

解析：模拟，带进去算分别是0,1,4,9,5,3,3,5，最后的3和5冲突了，要移动到6和7。

7. 答案：C

解析：又是一道原题，留一个点，其他两两有边，9∗(9−1)/2=369∗(9−1)/2=36

8. 答案：B

解析：根的节点为1，满足高度为k最多2k−12k−1个节点，211−1=2047211−1=2047

9. 答案：D

解析：前序遍历为根左右，中序遍历为左根右，根肯定要访问，只有左子树全部没有。

10. 答案：A

解析：模拟就可以，如果知道逆序对的性质直接算下逆序对。

11. 答案：A

解析：模拟的话我认为这个题非常麻烦，可以使用费马小定理。p为质数，满足ap−1≡1(mod)pap−1≡1(mod)p，那么solve(t,n)=5t−1modnsolve(t,n)=5t−1modn，即 a22≡1(mod)23a22≡1(mod)23

12. 答案：C

解析：每次递归分之为2次，n次需要分支，大约就是n个2相乘。

13. 答案：C

解析：这个题是列举不出来的，当前选的苹果可以在之前选or不选的基础上，可能很容易找到递推方程为斐波那契。但是至少选1个，把不选的减去。即求Fib(8)-1

14. 答案：C

解析：利用集合，先计算ab、bc、ac相等的情况个数，再减去abc相等的个数。如果腰的长度是x，底的长度必须小于2x，有1+3+5+7+9+9+9+9+9=61种，每个均有3种排列。等边三角形9个，会被多算两次。即3∗61−2∗9=1653∗61−2∗9=165

15. 答案：B

解析：这个题有层次，可以和数塔一样从后往前推。

二、阅读程序

1.几何相关，没学过高中的三角函数大概这题考的不高

1) 答案：T

解析：t算出来的是整数，没有经过取整。

2) 答案：F

解析：先/2向下取整了，如果是奇数不就没了。

3) 答案：F

解析：sq的参数会被转为int，但是x和y是double啊，有sqrt(2)

4) 答案：T

解析：acos(0.5)，高中的你大概接触过三角函数？这不就是PI/3吗，模拟快的。

5) 答案：D

解析：模拟

6) 答案：C

解析：分析单位，肯定是体积了。求出来的球心距离，距离和半径比，分别对应互相包含和不包含。如果搞不懂上面的话包含取min还是懂吧。

2.求最大子段和，slove1通过合并，slove2通过分治。

1) 答案：T

解析：n<=0有判断，正常执行满足。是不是想等的呢，得看程序了。

2) 答案：F

解析：n<=0时正常执行。

3) 答案：F

解析：这个题就要看懂程序了，如果没接触过这个算法，大概很受伤。

4) 答案：B

解析：等比数列求和，不超过n

5) 答案：C

解析：分治很多都是多个log，比如归并。这个程序是T(n)=2∗T(n/2)+nT(n)=2∗T(n/2)+n，赛前也推过的

6) 答案：B

解析：模拟算下就行，看懂程序很简单。

3.比赛前一天早上出了这一套题，还没来得及给学生练。TZOJ7214: Base64编码，如果了解背景这题就是送的。不了解的话由于有些位运算，可能不太好做，搞了一个table存了些字符，可以理解就是加密解密。

1) 答案：F

解析：decode过程中出现'\n'，就不是一行了

2) 答案：T

解析：编码解码就是这样的。

3) 答案：F

解析：得算一下。

4) 答案：B

解析：encode只有一个n的循环，string只有添加字符操作

5) 答案：D(我认为D或BD或全对)

解析：char C语言并没有定义是否unsigned，即ub(undefined behavior)。

6) 答案：D

解析：长度%3=1，两个==，然后模拟那么不一样的就可以了。

三、完善程序

1.搞了个类dij，每次满足条件都会update松弛下。F[i]表示到i需要几个4，vis表示是否算过。

1) 答案：D

解析：初始条件

2) 答案：A

解析：没有算过就算

3) 答案：D

解析：找到F中最小的数松弛

4) 答案：C

解析：没有算过才能操作。但是去掉也可以

2.先处理笛卡尔树，再DFS求出欧拉序，然后求块间的最小值，最后求块内的最小值。难飞了

1) 答案：A

解析：构建笛卡尔树，链接当前节点的左孩子。

2) 答案：D

解析：构建笛卡尔树，当前节点链接到断开位置的右孩子。

3) 答案：A

解析：dfs树上求LCA，这个简单，深度越浅越优。

4) 答案：D

解析：这个有点送，只有一个根据深度来的。

5) 答案：D

解析：这个题要用前一题，我不会

6) 答案：C

解析：89行返回值中，较低位是左边的，要右移，而且和l有关。