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
```