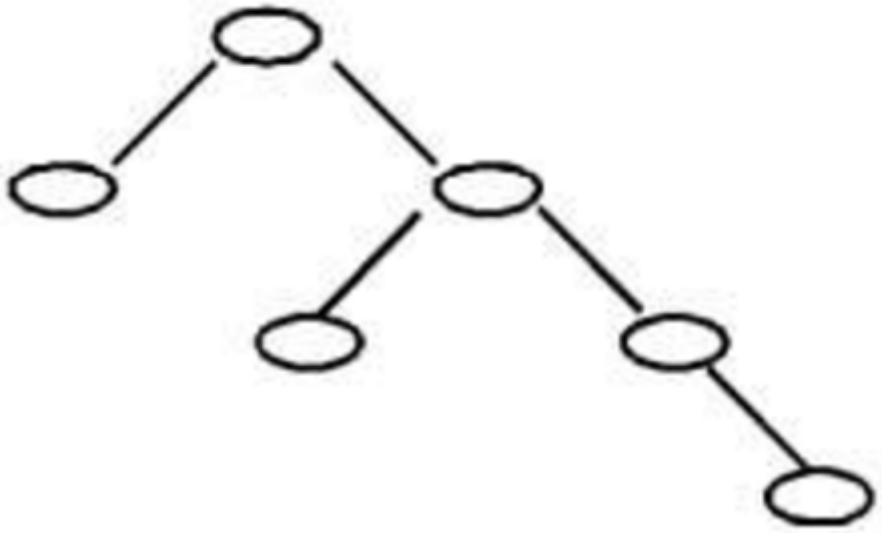
普及组初赛模拟题（十一）

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. 十进制算术表达式：$5 \times 512 + 7 \times 64 + 4 \times 8 + 7$，运算结果用二进制表示是：
   - A. `111111100101`
   - B. `111110100101`
   - C. `101111100111`
   - D. `111111011011`
2. 有 20 只猴子顺时针围成一圈，编号分别为 1 至 20，从 1 号猴子开始顺时针报数，数字从 1 开始数下去，1，2，3，4，$\cdots$，20，21，22，$\cdots$，一圈又一圈，当数到数字 x 时停止，报数字 x 猴子的编号是：
   - A. `(X - 1) % 20`
   - B. `(X + 1) % 20 - 1`
   - C. `1 + (X - 1) % 20`
   - D. `(X + 1) % 20`
3. 已知每个 integer 类型的变量需要用 2 个字节的空间存放，则数组 `integer a[10][2];` 需要占用的内存空间字节数是：
   - A. 80
   - B. 100
   - C. 200
   - D. 40
4. 设有 60 个已排好序的数据元素，采用折半查找时，最大比较次数为()。
   - A. 6
   - B. 9
   - C. 5
   - D. 7
5. 比 120 大的最小素数是()。
   - A. 125
   - B. 123
   - C. 127
   - D. 129
6. 由 4 个节点构成的形态不同的二叉树有（ ）种
   - A. 16

- B. 14
- C. 20
- D. 10

7. 已知大写字母 `A` 的 ASCII 编码为 65（十进制），则大写字母 `J` 的十进制 ASCII 编码为：
   - A. 71
   - B. 72
   - C. 73
   - D. 74
8. 小明有 n 块石头，现在小明要将其分为 a, b 两类石头，其中 a 类石头的数量必须是 4 的倍数，b 类石头的数量不超过 3 个，请问小明有多少种分类方法（ ）。
   - A. 1
   - B. n
   - C. $n \times n$
   - D. $\frac{n \times (n-1)}{2}$
9. 设 `a = false`，`b = true`，`c = true` ，以下表达式为假的是（）。
   - A.$(a \wedge b) \wedge c$
   - B.$(a \wedge b) \vee (b \wedge c)$
   - C.$(a \vee b) \wedge c$
   - D.$(a \vee c) \wedge (c \wedge b)$
10. 所有满足除以 3 的余数为 2，除以 7 的余数为 1 中，最小的数是 8，第五小的数是（）。
    - A. 40
    - B. 50
    - C. 71
    - D. 92
11. 一棵二叉树的前序遍历序列是 `ABCDEFG` .后序遍历序列是 `CBFEGDA` .则根结点的左子树的结点个数可能是() 。
    - A. 3
    - B. 5
    - C. 2
    - D. 4
12. 一棵二叉树如下图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点（根结点的下标为 1，若某结点的下标为 i，则其左孩子位于下标 $2i$ 处、右孩子位于下标 $2i+1$ 处），则该数组的最大下标至少为()。



- A. 6
- B. 10
- C. 12
- D. 15

13. 循环队列是一种连续储存数据的数据结构，已知循环队列空间为 30，队头位置编号为 12，队尾元素下一个空位置编号为 5，则队伍中元素个数为（ ）
    - A. 8
    - B. 7
    - C. 23
    - D. 22
14. 8 位二进制数中去掉符号位，最大能表示多少字符（ ）
    - A. 128
    - B. 127
    - C. 255
    - D. 256
15. 用某种排序方法对线性表 `25,84,21,47,15,27,68,35,20` 进行排序，结点变化如下:

- (1) `25,84,21,47,15,27,68,35,20` ；
- (2) `20,15,21,25,47,27,68,35,84` ；
- (3) `15,20,21,25,35,27,47,68,84` ；
- (4) `15,20,21,25,27,35,47,68,84` 。

那么，排序方法是（ ）

- A.希尔排序
- B.快速排序
- C.选择排序
- D.归并排序

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 T，错误填 F；除特殊说明外， 判断题 2 分，选择题 3 分，共计 40 分）

注意：判断题正确填 `T` ，错误填 `F` 。

阅读下面程序，完成第 $16 \sim 21$ 题。

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    string s;
    char s1[100];
    int len, j = 0;
    cin >> s;
    len = s.size();
    memset(s1, 0, sizeof(s1));
    for (int i = 0; i < len; i++) {
        if (i % 2 == 0)
            if ((s[i] >= 'A' && s[i] < 'Z') || (s[i] >= 'a' && s[i] < 'z')) {
                s1[j] = s[i] + 1;
                ++j;
            }
    }
    cout << s1 << endl;
    return 0;
}
```

16. (1.5 分)输出的字符串只能是字母组成。[ ]
17. (1.5 分)将 `s[i] < 'Z'` 的 `<` 改为 `<=` 则输出结果有可能包含数字。[ ]
18. 将 `memset(s1, 0, sizeof(s1));` 删除，程序运行结果不会改。[ ]
19. 将 `if (i % 2 == 0)` 删除，输出字符的长度和输入字符的长度一致。[ ]
20. 如输入的字符串长度为 10，则输出的字符串长度最长可能为多少（ ）
    - A. 4
    - B. 5
    - C. 10
    - D. 6
21. 如输入的字符串都是字母，则输出中哪个字母可能出现（ ）
    - A. `a`
    - B. `A`
    - C. `Z`
    - D.以上都不对

阅读下面程序，完成第 $22 \sim 27$ 题。

```cpp
#include<iostream>
using namespace std;
const int NUM = 5;
int r(int n) {
    int i;
    if (n <= NUM) return n;
    for (int i = 1; i <= NUM; ++i)
        if (r(n - i) < 0) return i;
    return -1;
}
int main() {
    int n;
    cin >> n;
    cout << r(n) << endl;
    return 0;
}
```

22. (1.5 分)将 `for (int i = 1; i <= NUM; ++i)` 的 `i=1` 改为 `i=0` ,程序不会出错。[ ]
23. 程序输出的结果有可能小于 -1。[ ]
24. 若输入的 n 大于等于 6 时，程序一定至少执行一次 `return -1;` 。[ ]
25. 若程序两次输入的值分别为 n_1 和 n_2，且有 $n_1 - n_2 = 1$ 的关系,则对于这两次运行的结果 ans_1 和 ans_2,有 $ans_1 - ans_2 = 1$。[ ]
26. 若已知 $0 \leq n \leq 100$，则要使输出的结果为 -1 则 n 的取值有()种。
    - A. 16
    - B. 14
    - C. 12
    - D. 10
27. 若输入 `2020` ,期望的输出的结果为()。
    - A. `-1`
    - B. `3`
    - C. `4`
    - D. `5`

阅读下面程序，完成第 $28 \sim 33$ 题。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int t, x[100], a[100];
void work(int d, int i, int n) {
    int k;
    if (n == 1) {
        for (k = 0; k < d; k++) {
            printf("%3d", a[k]);
        }
        printf("\n");
    } else {
        for (k = i; k < t; k++) {
            if (n % x[k] == 0) {
                a[d] = x[k];
                work(d + 1, k, n / x[k]);
            }
        }
    }
}
int main() {
    int i, k, n;
    cin >> n;
    for (i = n; i > 1; i--) {
        if (n % i == 0) {
            x[t++] = i;
        }
    }
    work(0, 0, n);
    return 0;
}
```

28. `for (int i = n; i > 1; i--) if (n % i == 0) x[t++] = i;` 的作用是求出 n 的所有因数。[ ]
29. 该程序的作用是对 n 进行质因数分解。[ ]
30. (1.5 分) `printf("%3d" ,a[k]);` 中去掉 3 对程序没有影响。[ ]
31. 去掉 `if(n%x[k]==0)` 对程序有影响。[ ]
32. 如果输入为 `2` ，那么输出为。( )
    - A. `2`
    - B. `2 1`
    - C. `1 2`
    - D. `2 2`
33. 如果输入为 `72` ，那么输出有多少非空行。( )
    - A. 14
    - B. 15
    - C. 16

- D. 17

三、完善程序（单选题，每小题3分，共计30分）

阅读下面题目，完成第 $34 \sim 38$ 题。

求 A 的 B 次方的最后三位数
输入: 两个正整数 A 和 B。
输出: A^B 的最后三位。

```
#include<cstdio>

using namespace std;

int FastExponentiation(int a, int b, int mod) {
    int answer = ___(1)___;
    while (b != 0) {
        if ( ___(2)___ ) {
            answer *= a;
            ___(3)___;
        }
        b /= 2;
        ___(4)___;
        a %= mod;
    }
    return answer;
}

int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d\n", ___(5)___ );
    return 0;
}
```

34. (1) 处应填（ ）。
    - A. `0`
    - B. `1`
    - C. `1000`
    - D. `-1`
35. (2) 处应填（ ）。
    - A. `b % 2 == 1`
    - B. `b % 2 == 0`
    - C. `b == 1`

- D. `b == 0`

36. (3) 处应填（ ）。
    - A. `answer *= mod`
    - B. `answer /= mod`
    - C. `answer %= mod`
    - D. `answer *= answer`
37. (4) 处应填（ ）。
    - A. `a *= a`
    - B. `a *= b`
    - C. `b *= a`
    - D. `b *= b`
38. (5) 处应填（ ）。
    - A. `FastExponentitation(b, a, 1000)`
    - B. `FastExponentitation(a, b, 1000)`
    - C. `FastExponentitation(b, a, 3)`
    - D. `FastExponentitation(a, b, 3)`

阅读下面题目，完成第 $39 \sim 43$ 题。

题目描述

在一个月黑风高的夜晚，有一群人在河的右岸，想通过唯一的一根独木桥走到河的左岸。在伸手不见五指的黑夜里，过桥时必须借照灯光来照明，不幸的是，他们只有一盏灯。另外，独木桥上最多能承受两个人同时经过，否则将会坍塌。每个人单独过独木桥都需要一定的时间，不同的人用的时间可能不同。两个人一起过独木桥时，由于只有一盏灯，所以需要的时间是较慢的那个人单独过桥所花费的时间。现在输入 $N(2 \leq N \leq 1000)$ 和这 N 个人单独过桥需要的时间，请计算总共最少需要多少时间，他们才能全部到达河左岸。

例如，有 3 个人甲、乙、丙，他们单独过桥的时间分别为 1、2、4，则总共最少需要的时间为 7。具体方法是:甲、乙一起过桥到河的左岸，甲单独回到河的右岸将灯带回，然后甲、丙在一起过桥到河的左岸，总时间为 $2+1+4=7$。

```cpp
#include<iostream>

#include<cstring>

using namespace std;
const int SIZE = 100;
const int INFINITY = 10000;
const bool LEFT = true;
const bool RIGHT = false;
const bool LEFT_TO_RIGHT = true;
const bool RIGHT_TO_LEFT = false;
int n, hour[SIZE];
bool pos[SIZE];
int max(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}
int go(bool stage) {
    int i, j, num, tmp, ans;
    if (stage == RIGHT_TO_LEFT) {
        num = 0;
        ans = 0;
        for (i = 1; i <= n; ++i)
            if (pos[i] == RIGHT) {
                num++;
                if (hour[i] > ans)
                    ans = hour[i];
            }
        if (___(1) ___)
            return ans;
        ans = INFINITY;
        for (i = 1; i <= n - 1; ++i)
            if (pos[i] == RIGHT)
                for (j = i + 1; j <= n; ++j)
                    if (pos[j] == RIGHT) {
                        pos[i] = LEFT;
                        pos[j] = LEFT;
                        tmp = max(hour[i], hour[j]) + ___(2) ___;
                        if (tmp < ans)
                            ans = tmp;
                        pos[i] = RIGHT;
                        pos[j] = RIGHT;
                    }
        return ans;
    }
    if (stage == LEFT_TO_ RIGHT) {
        ans = INFINITY;
        for (i = 1; i <= n; ++i)
```

```
            if (___(3) ___) {
                pos[i] = RIGHT;
                tmp = ___(4) ___;
                if (tmp < ans)
                    ans = tmp;
                ___(5) ___;
            }
        return ans;
    }
    return 0;
}
int main() {
    int i;
    cin >> n;
    for (i = 1; i <= n; ++i) {
        cin >> hour[i];
        pos[i] = RIGHT;
    }
    cout << go(RIGHT_TO_LEFT) << endl;
    return 0;
}
```

1. ①处应填( )
   - A. `num <= 1`
   - B. `num <= 3`
   - C. `num <= 2`
   - D. `num <= 0`
2. ②处应填( )
   - A. `go(RIGHT)`
   - B. `go(!pos[i])`
   - C. `go(RIGHT_TO_LEFT)`
   - D. `go(LEFT_TO_RIGHT)`
3. ③处应填( )
   - A. `pos[i] == LEFT`
   - B. `num <= 3`
   - C. `pos[i] == RIGHT`
   - D. `num <= 2`
4. ④处应填( )
   - A. `go(RIGHT_TO_LEFT)`
   - B. `go(LEFT_TO_RIGHT)`
   - C. `hour[i] + go(RIGHT_TO_LEFT)`
   - D. `hour[i] + go(LEFT_TO_RIGHT)`
5. ⑤处应填( )

- A. `pos[i] = RIGHT`
- B. `pos[i] = LEFT`
- C. `tmp = 0`
- D. `return ans`