第二十二届全国青少年信息学奥林匹克联赛初赛

提高组 C++语言试题

竞赛时间：2016 年 10 月 22 日 14:30~16:30

选手注意：

- 试题纸共有 13 页，答题纸共有 2 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 1.5 分，共计 22.5 分；每题有且仅有一个正确选项）

1. 以下不是微软公司出品的软件是（　）。

A. Powerpoint　　B. Word
C. Excel　　D. Acrobat Reader

2. 如果开始时计算机处于小写输入状态，现在有一只小老鼠反复按照 CapsLock、字母键 A、字母键 S 和字母键 D 的顺序来回按键，即 CapsLock、A、S、D、S、A、CapsLock、A、S、D、S、A、CapsLock、A、S、D、S、A、……，屏幕上输出的第 81 个字符是字母（　）。

A. A　　B. S　　C. D　　D. a

3. 二进制数 00101100 和 01010101 异或的结果是（　）。

A. 00101000　　B. 01111001　　C. 01000100　　D. 00111000

4. 与二进制小数 0.1 相等的八进进制数是（　）。

A. 0.8　　B. 0.4　　C. 0.2　　D. 0.1

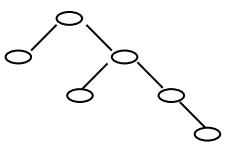
5. 以比较作为基本运算，在 N 个数中找最小数的最少运算次数为（　）。

A. N　　B. N-1　　C. N^2　　D. log N

6. 表达式 a*(b+c)-d 的后缀表达形式为（　）。

A. abcd*+-　　B. abc+*d-　　C. abc*+d-　　D. -+*abcd

7. 一棵二叉树如右图所示，若采用二叉树链表存储该二叉树（各个结点包括结点的数据、左孩子指针、右孩子指针）。如果没有左孩子或者右孩子，则对应的为空指针。那么该链表中空指针的数目为（　）。



A. 6　　B. 7　　C. 12　　D. 14

8. G 是一个非连通简单无向图，共有 28 条边，则该图至少有（　　）个顶点。

A. 10　　B. 9　　C. 8　　D. 7

9. 某计算机的 CPU 和内存之间的地址总线宽度是 32 位（bit），这台计算机最多可以使用（　　）的内存。

A. 2GB　　B. 4GB　　C. 8GB　　D. 16GB

10. 有以下程序：

```
#include <iostream>
using namespace std;

int main() {
    int k = 4, n = 0;
    while (n < k) {
        n++;
        if (n % 3 != 0)
            continue;
        k--;
    }
    cout << k << "," << n << endl;
    return 0;
}
```

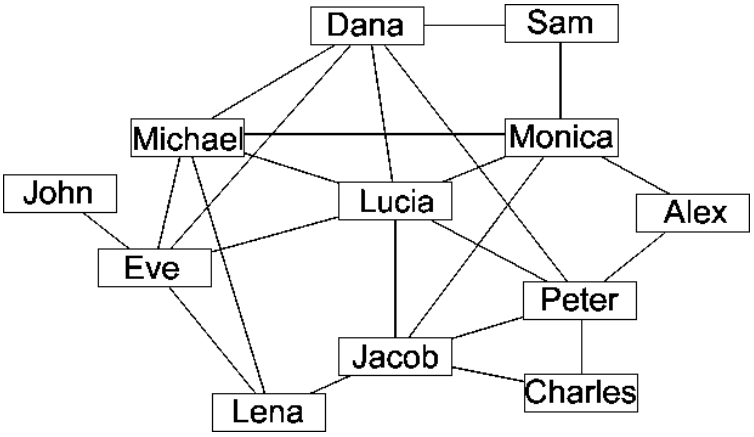
程序运行后的输出结果是（　　）。

A. 2,2　　B. 2,3　　C. 3,2　　D. 3,3

11. 有 7 个一模一样的苹果，放到 3 个一样的盘子中，一共有（　　）种放法。

A. 7　　B. 8　　C. 21　　D. 3^7

12. Lucia 和她的朋友以及朋友的朋友都在某社交网站上注册了账号。下图是他们之间的关系图，两个人之间有边相连代表这两个人是朋友，没有边相连代表不是朋友。这个社交网站的规则是：如果某人 A 向他（她）的朋友 B 分享了某张照片，那么 B 就可以对该照片进行评论；如果 B 评论了该照片，那么他（她）的所有朋友都可以看见这个评论以及被评论的照片，但是不能对该照



片进行评论（除非 A 也向他（她）分享了该照片）。现在 Lucia 已经上传了一张照片，但是她不想让 Jacob 看见这张照片，那么她可以向以下朋友（　　）分享该照片。

A. Dana, Michael, Eve　　B. Dana, Eve, Monica
C. Michael, Eve, Jacob　　D. Micheal, Peter, Monica

13. 周末小明和爸爸妈妈三个人一起想动手做三道菜。小明负责洗菜、爸爸负责切菜、妈妈负责炒菜。假设做每道菜的顺序都是：先洗菜 10 分钟，然后切菜 10 分钟，最后炒菜 10 分钟。那么做一道菜需要 30 分钟。注意：两道不同的菜的相同步骤不可以同时进行。例如第一道菜和第二道的菜不能同时洗，也不能同时切。那么做完三道菜的最短时间需要（　　）分钟。

A. 90　　B. 60　　C. 50　　D. 40

14. 假设某算法的计算时间表示为递推关系式

$$T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$$

$$T(1) = 1$$

则算法的时间复杂度为（　　）。

A. $O(n)$　　B. $O(\sqrt{n})$　　C. $O(\sqrt{n}\log n)$　　D. $O(n^2)$

15. 给定含有 n 个不同的数的数组 $L=<x_1, x_2, ..., x_n>$。如果 L 中存在 x_i（$1 < i < n$）使得 $x_1 < x_2 < ... < x_{i-1} < x_i > x_{i+1} > ... > x_n$，　则称 L 是单峰的，并称 x_i 是 L 的“峰顶”。现在已知 L 是单峰的，请把 a-c 三行代码补全到算法中使得算法正确找到 L 的峰顶。

```
a. Search(k+1, n)
b. Search(1, k-1)
c. return L[k]

Search(1, n)
1. k←⌊n/2⌋
2. if L[k] > L[k-1] and L[k] > L[k+1]
3. then __________
4. else if L[k] > L[k-1] and L[k] < L[k+1]
5. then __________
6. else __________
```

正确的填空顺序是（　　）。

A. c, a, b　　B. c, b, a　　C. a, b, c　　D. b, a, c

二、不定项选择题（共 5 题，每题 1.5 分，共计 7.5 分；每题有一个或多个正确选项，多选或少选均不得分）

1. 以下属于无线通信技术的有（　　）。

A. 蓝牙　　B. WiFi　　C. GPRS　　D. 以太网

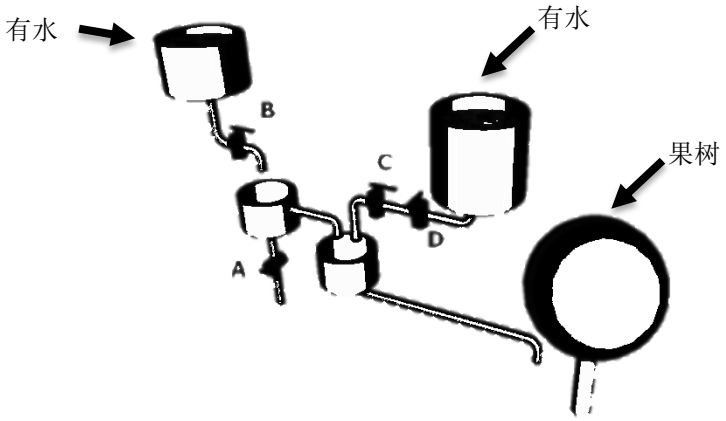
2. 可以将单个计算机接入到计算机网络中的网络接入通讯设备有（　　）。

A. 网卡　　B. 光驱　　C. 鼠标　　D. 显卡

3. 下列算法中运用分治思想的有（　　）。

A. 快速排序　　B. 归并排序　　C. 冒泡排序　　D. 计数排序

4. 下图表示一个果园灌溉系统，有 A、B、C、D 四个阀门，每个阀门可以打开或关上，所有管道粗细相同，以下设置阀门的方法中，可以让果树浇上水的有（　　）。



A. B 打开，其他都关上　　B. AB 都打开，CD 都关上

C. A 打开，其他都关上　　D. D 打开，其他都关上

5. 参加 NOI 比赛，以下能带入考场的有（　　）。

A. 钢笔　　B. 适量的衣服　　C. U 盘　　D. 铅笔

三、问题求解（共 2 题，每题 5 分，共计 10 分；每题全部答对得 5 分，没有部分分）

1. 一个 1×8 的方格图形（不可旋转）用黑、白两种颜色填涂每个方格。如果每个方格只能填涂一种颜色，且不允许两个黑格相邻，共有_____种填涂方案。

2. 某中学在安排期末考试时发现，有 7 个学生要参加 7 门课程的考试，下表列出了哪些学生参加哪些考试（用 √ 表示要参加相应的考试）。

最少要安排_____个不同的考试时间段才能避免冲突？

考试	学生 1	学生 2	学生 3	学生 4	学生 5	学生 6	学生 7
通用技术	√				√		√
物理	√	√					√
化学		√		√			
生物	√				√	√	
历史			√	√		√	
地理		√	√				√
政治			√			√	

四、阅读程序写结果（共 4 题，每题 8 分，共计 32 分）

1.
```
#include <iostream>
using namespace std;

int main() {
    int a[6] = {1, 2, 3, 4, 5, 6};
    int pi = 0;
    int pj = 5;
    int t , i;
    while (pi < pj) {
        t = a[pi];
        a[pi] = a[pj];
        a[pj] = t;
        pi++;
        pj--;
    }
    for (i = 0; i < 6; i++)
        cout << a[i] << ",";
    cout << endl;
    return 0;
}
```

输出：_____

2.
```
#include <iostream>
using namespace std;

int main() {
    char a[100][100], b[100][100];
    string c[100];
    string tmp;
    int n, i = 0, j = 0, k = 0, total_len[100], length[100][3];
```

```
cin >> n;
getline(cin, tmp);
for (i = 0; i < n; i++) {
    getline(cin, c[i]);
    total_len[i] = c[i].size();
}
for (i = 0; i < n; i++) {
    j = 0;
    while (c[i][j] != ':') {
        a[i][k] = c[i][j];
        k = k + 1;
        j++;
    }
    length[i][1] = k - 1;
    a[i][k] = 0;
    k = 0;
    for (j = j + 1; j < total_len[i]; j++) {
        b[i][k] = c[i][j];
        k = k + 1;
    }
    length[i][2] = k - 1;
    b[i][k] = 0;
    k = 0;
}
for (i = 0; i < n; i++) {
    if (length[i][1] >= length[i][2])
        cout << "NO,";
    else {
        k = 0;
        for (j = 0; j < length[i][2]; j++) {
            if (a[i][k] == b[i][j])
                k = k + 1;
            if (k > length[i][1])
                break;
        }
        if (j == length[i][2])
            cout << "NO,";
        else
            cout << "YES,";
    }
}
cout << endl;
return 0;
```

```
}
```

输入：3
AB:ACDEbFBkBD
AR:ACDBrT
SARS:Severe Atypical Respiratory Syndrome
输出：_____
（注：输入各行前后均无空格）

3.
```
#include <iostream>
using namespace std;

int lps(string seq, int i, int j) {
    int len1, len2;
    if (i == j)
        return 1;
    if (i > j)
        return 0;
    if (seq[i] == seq[j])
        return lps(seq, i + 1, j - 1) + 2;
    len1 = lps(seq, i, j - 1);
    len2 = lps(seq, i + 1, j);
    if (len1 > len2)
        return len1;
    return len2;
}

int main() {
    string seq = "acmerandacm";
    int n = seq.size();
    cout << lps(seq, 0, n - 1) << endl;
    return 0;
}
```

输出：_____

4.
```
#include <iostream>
#include <cstring>
using namespace std;

int map[100][100];
int sum[100], weight[100];
int visit[100];
```

```
int n;

void dfs(int node) {
    visit[node] = 1;
    sum[node] = 1;
    int v, maxw = 0;
    for (v = 1; v <= n; v++) {
        if (!map[node][v] || visit[v])
            continue;
        dfs(v);
        sum[node] += sum[v];
        if (sum[v] > maxw)
            maxw = sum[v];
    }
    if (n - sum[node] > maxw)
        maxw = n - sum[node];
    weight[node] = maxw;
}

int main() {
    memset(map, 0, sizeof(map));
    memset(sum, 0, sizeof(sum));
    memset(weight, 0, sizeof(weight));
    memset(visit, 0, sizeof(visit));
    cin >> n;
    int i, x, y;
    for (i = 1; i < n; i++) {
        cin >> x >> y;
        map[x][y] = 1;
        map[y][x] = 1;
    }
    dfs(1);
    int ans = n, ansN = 0;
    for (i = 1; i <= n; i++)
        if (weight[i] < ans) {
            ans = weight[i];
            ansN = i;
        }
    cout << ansN << " " << ans << endl;
    return 0;
}
```

输入：11

```
1 2
1 3
2 4
2 5
2 6
3 7
7 8
7 11
6 9
9 10
```

输出：_____

五、完善程序（共 2 题，每题 14 分，共计 28 分）

1. （交朋友）根据社会学研究表明，人们都喜欢找和自己身高相近的人做朋友。现在有 n 名身高两两不相同的同学依次走入教室，调查人员想预测每个人在走入教室的瞬间最想和已经进入教室的哪个人做朋友。当有两名同学和这名同学的身高差一样时，这名同学会更想和高的那个人做朋友。比如一名身高为 1.80 米的同学进入教室时，有一名身高为 1.79 米的同学和一名身高为 1.81 米的同学在教室里，那么这名身高为 1.80 米的同学会更想和身高为 1.81 米的同学做朋友。对于第一个走入教室的同学我们不做预测。

由于我们知道所有人的身高和走进教室的次序，所以我们可以采用离线的做法来解决这样的问题，我们用排序加链表的方式帮助每一个人找到在他之前进入教室的并且和他身高最相近的人。（第一空 2 分，其余 3 分）

```
#include <iostream>
using namespace std;
#define MAXN 200000
#define infinity 2147483647

int answer[MAXN], height[MAXN], previous[MAXN], next[MAXN];
int rank[MAXN];
int n;

void sort(int l, int r) {
   int x = height[rank[(l + r) / 2]], i = l, j = r, temp;
   while (i <= j)
   {
      while (height[rank[i]] < x) i++;
      while (height[rank[j]] > x) j--;
      if (    (1)    ) {
         temp = rank[i]; rank[i] = rank[j]; rank[j] = temp;
```

```
            i++; j--;
        }
    }
    if (i < r) sort(i, r);
    if (l < j) sort(l, j);
}

int main()
{
    cin >> n;
    int i, higher, shorter;
    for (i = 1; i <= n; i++) {
        cin >> height[i];
        rank[i] = i;
    }
    sort(1, n);
    for (i = 1; i <= n; i++) {
        previous[rank[i]] = rank[i - 1];
        ___(2)___;
    }
    for (i = n; i >= 2; i--) {
        higher = shorter = infinity;
        if (previous[i] != 0)
            shorter = height[i] - height[previous[i]];
        if (next[i] != 0)
            ___(3)___;
        if (___(4)___)
            answer[i] = previous[i];
        else
            answer[i] = next[i];
        next[previous[i]] = next[i];
        ___(5)___;
    }
    for (i = 2; i <= n; i++)
        cout << i << ":" << answer[i];
    return 0;
}
```

2. （**交通中断**）有一个小国家，国家内有 n 座城市和 m 条双向的道路，每条道路连接着两座不同的城市。其中 1 号城市为国家的首都。由于地震频繁可能导致某一个城市与外界交通全部中断。这个国家的首脑想知道，如果只有第 i(i>1)个城市因地震而导致交通中断时，首都到多少个城市的最短路径长度会发生改变。如果因为无法通过第 i 个城市而导致从首都出发无法到达某个城

市，也认为到达该城市的最短路径长度改变。

对于每一个城市 i，假定只有第 i 个城市与外界交通中断，输出有多少个城市会因此导致到首都的最短路径长度改变。

我们采用邻接表的方式存储图的信息，其中 head[x]表示顶点 x 的第一条边的编号，next[i]表示第 i 条边的下一条边的编号，point[i]表示第 i 条边的终点，weight[i]表示第 i 条边的长度。（第一空 2 分，其余 3 分）

```
#include <iostream>
#include <cstring>
using namespace std;
#define MAXN 6000
#define MAXM 100000
#define infinity 2147483647

int head[MAXN], next[MAXM], point[MAXM], weight[MAXM];
int queue[MAXN], dist[MAXN], visit[MAXN];
int n, m, x, y, z, total = 0, answer;

void link(int x,int y,int z) {
   total++;
   next[total] = head[x];
   head[x] = total;
   point[total] = y;
   weight[total] = z;
   total++;
   next[total] = head[y];
   head[y] = total;
   point[total] = x;
   weight[total] = z;
}

int main() {
    int i, j, s, t;
   cin >> n >> m;
   for (i = 1; i <= m; i++) {
      cin >> x >> y >> z;
      link(x, y, z);
   }
   for (i = 1; i <= n; i++) dist[i] = infinity;
       (1)    ;
   queue[1] = 1;
   visit[1] = 1;
   s = 1;
```

```
    t = 1;
    // 使用 SPFA 求出第一个点到其余各点的最短路长度
    while (s <= t) {
        x = queue[s % MAXN];
        j = head[x];
        while (j != 0) {
            if (    (2)    ) {
                dist[point[j]] = dist[x] + weight[j];
                if (visit[point[j]] == 0) {
                    t++;
                    queue[t % MAXN] = point[j];
                    visit[point[j]] = 1;
                }
            }
            j = next[j];
        }
            (3)    ;
        s++;
    }
    for (i = 2; i <= n; i++) {
        queue[1] = 1;
        memset(visit, 0, sizeof(visit));
        visit[1] = 1;
        s = 1;
        t = 1;
        while (s <= t) { // 判断最短路长度是否不变
            x = queue[s];
            j = head[x];
            while (j != 0) {
                if (point[j] != i &&     (4)
                        && visit[point[j]] == 0) {
                     (5)    ;
                    t++;
                    queue[t] = point[j];
                }
                j = next[j];
            }
            s++;
        }
        answer = 0;
        for (j = 1; j <= n; j++)
            answer += 1 - visit[j];
         cout << i << ":" << answer - 1 << endl;
    }
    return 0;
}
```