# CSP-S 2020 模拟试卷（7）

（提高组　C++ 语言试题　两小时完成）

**考生注意事项：**

- 试题共 9 页，答题纸共 1 页，满分 100 分。请在答题纸上作答，写在试题纸上一律无效。
- 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；每小题仅有一个正确答案）

1. 与十进制数 17.5625 相对应的八进制数是 （　　）

(A) 21.5625　(B) 21.44　(C) 21.73　(D) 21.731

2. -128 的补码表示为 （　　）

(A) 00000000　(B) 00000001　(C) 10000000　(D) 11111111

3. 以下不属于 TCP 拥塞控制算法的是 （　　）

(A) 慢启动　(B) 拥塞避免　(C) 快启动　(D) 快速重传

4. 以下不是基于 UDP 协议的是 （　　）

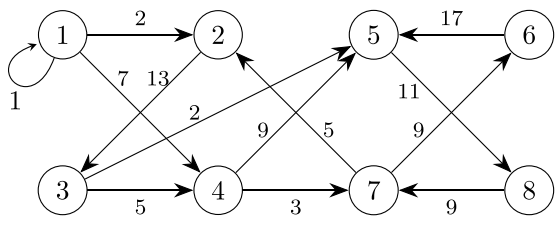
(A) DNS　(B) RIP　(C) TELNET　(D) TFTP

5. 定义如下函数 add_edge 和全局变量

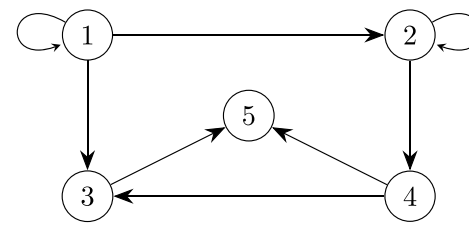
```
int to[MAX],nxt[MAX],h[MAX],top;
void add_edge(int u, int v){to[++top]=v, nxt[top]=h[u], h[u]=top;}
```

如左图节点编号从 1 开始，按边的编号顺序，以前向星的方式存储，请问 nxt[h[3]] 的值为 （　　）

(A) 6　(B) 3　(C) 8　(D) 7




第 5 题图




第 6 题图

6. 如右图所示，从节点 1 走 6 步走到节点 5 的方案数有多少种 （　　）

(A) 5　(B) 8　(C) 7　(D) 6

7. 循环队列数组的下标范围是 $0 \cdots m\text{-}1$，头尾指针分别为 $f$ 和 $r$，则队列中元素个数为 （　　）

(A) $r-f$　(B) $r-f+1$　(C) $r-f+1 \mod m$　(D) $r-f+m \mod m$

8. 将 2 个相同的红球，1 个蓝球，1 个白球放到 10 个编号不同的盒子中去，每个盒子最多放一个球，不同的放法数是 （ ）

(A) 5040 (B) 2520 (C) 420 (D) 1260

9. $G$ 是一个非连通简单无向图，共有 36 条边，则该图至少有几个顶点 （ ）

(A) 10 (B) 9 (C) 8 (D) 7

10. 由四个不同的点构成的简单无向连通图的个数是 （ ）

(A) 32 (B) 35 (C) 38 (D) 31

11. 前缀表达式- + * 4 + 2 3 1 5 的值为 （ ）

(A) 16 (B) 17 (C) 19 (D) 15

12. 2+3*(4-(5+6))/7 的逆波兰表达式为 （ ）

(A) 23456 − + ∗ 7/+ (B) 23456 − + ∗ /7+ (C) 23456 + − ∗ 7/+ (D) 23456 + + ∗ /7−

13. 有两个算法的时间计算递推关系分别是：$T(n) = 2.5T(2n/5) + n\log_2^2 n$ 和 $T'(n) = 3T'(n/4) + n\log_2 n$，则它们的算法复杂度分别为 （ ）

(A) $O(n\log_2^2 n)$, $O(nlog_2 n)$ (B) $O(n\log_2 n)$, $O(n)$

(C) $O(n\log_2^2 n)$, $O(n\log_2^2 n)$ (D) $O(n\log_2^3 n)$, $O(n\log_2 n)$

14. 两个人轮流抛硬币，最先抛出正面的人可以吃苹果，请问先抛的人能吃到苹果的概率是 （ ）

(A) $\frac{1}{2}$ (B) $\frac{2}{3}$ (C) $\frac{3}{4}$ (D) $\frac{5}{8}$

15. 有朋自远方来，他乘火车，轮船，汽车，飞机的概率分别为 0.3，0.2，0.1，0.4。如果他乘火车，轮船，汽车晚点到的概率分别为 1/4，1/3，1/12，而乘飞机来则不会晚点。下列说法错误的是 （ ）

(A) 坐陆路（火车，汽车）交通工具准点机会比坐水路（轮船）要高

(B) 如果他迟到，乘火车的概率是 0.5

(C) 如果他准点，坐轮船或汽车的概率等于坐火车的概率

(D) 如果他准点，那么乘飞机的概率大于等于 0.5

## 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 **2** 分，选择题 **3** 分，共计 **40** 分）

1.

```
#include<iostream>
using namespace std;
int a, b, c;

int* cal(int *p, int &q, int r) {
    q += r;
    *p += q;
    return p;
}

int main() {
    cin >> a >> b >> c;
```

```
    c = *cal(&a, b, c);
    cout << a << " " << b << " " << c;
    return 0;
}
```

- **判断题**

(1) cal 函数中参数 p 使用指针传递，q 和 r 则是值传递（　　）

(2) 第 13 行也可以用如下方式调用：`c = cal(&a, b, c)`（　　）

- **选择题**

(5) 当输入 1 2 3 时，程序输出结果为（　　）

(A) 6 2 3　　(B) 6 5 3　　(C) 6 5 6　　(D) 1 2 6

(6) 若输入 23 45 11，则输出是（　　）

(A) 79 56 11　　(B) 79 56 79　　(C) 44 56 79　　(D) 79 56 44

2.

```
#include<iostream>
#include<cmath>
#define MAX 1000
#define p sqrt(3)
using namespace std;
int n, dp[1000][3];
int h0 = 1, h1 = 3;
double ans1 = (2+p)/(2*p), ans2 = (-2+p)/(2*p);

int main() {
    cin >> n;
    dp[1][0] = dp[1][1] = dp[1][2] = 1;
    for (int i = 2, tmp; i <= n; i++) {
        dp[i][0] = dp[i-1][1] + dp[i-1][2];
        dp[i][1] = dp[i-1][0] + dp[i-1][1] + dp[i-1][2];
        dp[i][2] = dp[i-1][0] + dp[i-1][1] + dp[i-1][2];
        tmp = h1;
        h1 = 2*(h1+h0);
        h0 = tmp;
    }
    for(int i = 1; i <= n; i++) {
        ans1 = ans1*(1+p);
        ans2 = ans2*(1-p);
    }
    cout << h1 << endl;
    cout << dp[n][0] + dp[n][1] + dp[n][2] << endl;
    cout << ans1 + ans2 << endl;
}
```

- **判断题**

(1) 上述程序的输出中 h1 和 dp[n][0]+dp[n][1]+dp[n][2] 的值相等（　　）

(2) 上述程序的输出中 dp[n][0]+dp[n][1]+dp[n][2] 和 ans1+ans2 的值相等（　　）

- **选择题**

(5) 当 $n$ 等于 5 时，第一行输出 (即 $h1$) 结果为　　（　　）

(A) 164　　(B) 60　　(C) 448　　(D) 128

(6) 当 $n$ 等于 10 时，第三行输出 (即 ans1+ans2) 结果为　　（　　）

(A) 9136　　(B) 68192　　(C) 24960　　(D) 3344

3.

```
#include<iostream>
#include<cstring>
#define LL long long
using namespace std;
LL l, r;
LL f[12][10][10][2][2][2], a[20];

LL Dfs(LL now, LL p, LL pp, LL _4, LL _8, LL top, LL hw) {
    if (_4 && _8) return 0;
    if (!now) return hw;
    if (!top && f[now][p][pp][_4][_8][hw] != -1)
        return f[now][p][pp][_4][_8][hw];
    LL Up = top ? a[now] : 9;
    LL ret(0);
    for (LL i = 0; i <= Up; ++i)
        ret += Dfs(now-1, i, p, _4|(i==4), _8|(i==8),
                        top&&(i==Up), hw|(i==pp&&i==p));
    if (!top)
        f[now][p][pp][_4][_8][hw] = ret;
    return ret;
}
inline LL Solve(LL x) {
    LL tot(0);
    while (x) {
        a[++tot] = x%10;
        x /= 10;
    }
    if (tot != 11) return 0;
    LL ret(0);
    for (LL i = 1; i <= a[tot]; ++i)
        ret += Dfs(tot-1, i, 0, (i==4), (i==8), i==a[tot], 0);
    return ret;
}
int main() {
    cin >> l >> r;
    memset(f, -1, sizeof(f));
    cout << Solve(r)-Solve(l-1);
    return 0;
}
```

- **判断题**

(1) 同时包含 4 和 8 的数字都不会被统计（　　）

(2) 相邻数位中，超过 3 个数位相同的数字都不会被统计（　　）

- **选择题**

(5) 下列哪个是合法 (会被统计) 的数字　　（　　）

(A) 2323234823　　(B) 1015400080　　(C) 23333333333　　(D) 10010012022

(6) 当输入 12121284000 12121285550 时，程序输出结果为　　（　　）

(A) 5　　(B) 457　　(C) 455　　(D) 6

4.

```cpp
#include <iostream>
#include <string>

using namespace std;

size_t equalizeLength(string &s1, string &s2) {
    size_t len1 = s1.size(), len2 = s2.size();
    if (len1 < len2) {
        for (int i = 0; i < len2 - len1; ++i) s1 = '0' + s1;
        return len2;
    } else if (len1 > len2) {
        for (int i = 0; i < len1 - len2; ++i) s2 = '0' + s2;
    }
    return len1;
}

string strAddition(string s1, string s2) {
    string ret;
    int carry = 0;
    size_t len = equalizeLength(s1, s2);

    for (int i = len - 1; i >= 0; --i) {
        int firstBit = s1.at(i) - '0';
        int secondBit = s2.at(i) - '0';

        int sum = (firstBit ^ secondBit ^ carry) + '0';
        ret = static_cast<char>(sum) + ret;

        carry = (firstBit & secondBit) |
                (firstBit & carry) |
                (secondBit & carry);
    }
    if (carry)
        ret = '1' + ret;
    return ret;
}

long int Karatsuba(string s1, string s2) {
    size_t len = equalizeLength(s1, s2);
```

```
    // base case
    if (len == 0) return 0;
    if (len == 1) return (s1[0] - '0') * (s2[0] - '0');

    size_t floor = len / 2;
    size_t ceil = len - floor;
    string a = s1.substr(0, floor);
    string b = s1.substr(floor, ceil);
    string c = s2.substr(0, floor);
    string d = s2.substr(floor, ceil);

    long int p1 = Karatsuba(a, c);
    long int p2 = Karatsuba(b, d);
    long int p3 = Karatsuba(strAddition(a, b), strAddition(c, d));
    return (1<<(2 * ceil)) * p1 + (1<<(ceil)) * (p3 - p1 - p2) + p2;
}

int main() {
    string s1, s2;
    cin >> s1 >> s2;
    cout << Karatsuba(s1, s2) << endl;
    return 0;
}
```

- **判断题**

(1) 上述程序实现了大整数加法（　　）

(2) 上述程序的算法复杂度大于 $O(n^2)$（其中 $n$ 为 max(s1.length(),s2.length())）（　　）

- **选择题**

(5) 当输入 111 011 时程序输出为（　　）

(A) 10　　(B) 4　　(C) 21　　(D) 2

(6) 当输入 10101 101010 时程序输出为（　　）

(A) 441　　(B) 882　　(C) 1764　　(D) 220

## 三. 完善程序（单选题，每题 3 分，共计 30 分）

1.（链表反转）单向链表反转是一道经典算法问题，比如有一个链表是这样的，1->2->3->4->5，反转后成为 5->4->3->2->1。现给定如下链表节点的定义：

```
struct LinkNode {
    int value;
    LinkNode* next;
};
```

非递归实现：

```
LinkNode* Reverse(LinkNode* header) {
    if (header == NULL || header->next == NULL) {
```

```
        return header;
    }

    LinkNode* pre = header, *cur = header->next;
    pre->next = NULL;
    while(cur != NULL)
    {
        LinkNode* next = ____①____;
        ____②____ = pre;
        pre = cur;
        cur = next;
    }
    return pre;
}
```

递归实现：

```
LinkNode * Reverse(LinkNode * head) {
    if (head == NULL || head->next == NULL) {
        return head;
    }
    LinkNode * newhead = ____③____;

    ____④____ = head;
    head->next = ____⑤____;
    return newhead;
}
```

(1) ①处应填 ( )

(A) `pre->next` (B) `cur->next` (C) `header->next` (D) `NULL`

(2) ②处应填 ( )

(A) `pre->next` (B) `cur->next` (C) `header->next` (D) `NULL`

(3) ③处应填 ( )

(A) `ReverseList(head)` (B) `ReverseList(pre)`

(C) `ReverseList(cur)` (D) `ReverseList(head->next)`

(4) ④处应填 ( )

(A) `pre->next->next` (B) `cur->next->next`

(C) `header->next->next` (D) `NULL`

(5) ⑤处应填 ( )

(A) `pre->next` (B) `cur->next` (C) `header->next` (D) `NULL`

2. (最小环问题) 给定一张无向图，求图中一个至少包含 3 个点的环，环上的节点不重复，并且环上的边的长度之和最小。该问题称为无向图的最小环问题。在本题中，你需要输出最小环的方案，若最小环不唯一，输出任意一个均可。若无解，输出“No solution.”，图的节点数不超过 100100。

【输入】：第一行两个正整数 $n, m$ 表示点数和边数。接下来 $m$ 行，每行三个正整数 $x, y, z$，表示节点 $x, y$ 之间有一条长度为 $z$ 的边。

【输出】：一个最小环的方案：按环上顺序输出最小环上的点。若最小环不唯一，输出任意一个均可。若无解，输出 No solution.

```
#include <bits/stdc++.h>
#define MAXN 105
#define INF 0x3f3f3f3f
using namespace std;
inline int read() {
    int x = 0, f = 1;
    char ch = getchar();
    while (ch<'0' || ch>'9') {
        if (ch=='-') f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x<<3) + (x<<1) + (ch^'0');
        ch = getchar();
    }
    return x*f;
}
static int stk[MAXN], top;
static int pos[MAXN][MAXN]; //表示 i j 的中点节点 #define Push(x) stk[++top]=(x);
void GetAns(int i,int j) {
    if (pos[i][j] == 0) return ;
    GetAns(i, _____①_____);
    Push(pos[i][j]);
    GetAns(pos[i][j], _____②_____);
}
static int G[MAXN][MAXN],D[MAXN][MAXN];
int main() {
    int n=read(), m=read();
    memset(G,0x3f,sizeof(G));
    memset(D,0x3f,sizeof(D));
    for (register int i = 1; i <= m; ++i) {
        int u=read(), v=read();
        D[v][u]=D[u][v]=G[u][v]=G[v][u]=min(G[u][v],read());
    }
    int ans = INF;
    for (register int k=1; k<=n; ++k) {
        for (register int i=1; i<k; ++i) {
            for (register int j=i+1; j<k; ++j) {
                if (D[i][j]==INF||G[j][k]==INF||G[k][i]==INF) continue;
                if (D[i][j]+G[j][k]+G[k][i]<ans) {
                    ans = _____③_____;
                    top = 0;
                    Push(i);
                    GetAns(i,j);
                    Push(j);
```

```
                    Push(k);
                }
            }
        }
        for (register int i=1; i<=n; ++i) {
            for (register int j=1; j<=n; ++j) {
                if (          ④          ) {
                    D[i][j] = D[i][k]+D[k][j];
                    pos[i][j] = k;
                }
            }
        }
    }
    if (ans == INF) return puts("No solution."),0;
    for (register int i = 1; i <= top; ++i)
        printf("%d ", _____⑤_____);
}
```

(1) ①处应填 ( )

(A) `j` (B) `pos[i][j]` (C) `[i]` (D) `pos[j][i]`

(2) （2 分） ②处应填 ( )

(A) `j` (B) `pos[i][j]` (C) `[i]` (D) `pos[j][i]`

(3) （2 分） ③处应填 ( )

(A) `D[i][j]+G[k][j]+G[i][k]` (B) `D[i][j]+G[j][k]+G[k][i]`

(C) `D[i][k]+G[k][j]+G[i][j]` (D) `D[i][j]+G[j][i]+G[i][k]`

(4) （2 分） ④处应填 ( )

(A) `D[k][j]>D[i][k]+D[k][j]` (B) `D[i][j]>D[i][k]+D[k][j]`

(C) `D[i][j]<D[i][k]+D[k][j]` (D) `D[i][k]>D[i][k]+D[k][j]`

(5) ⑤处应填 ( )

(A) `pos[i][i]` (B) `stk[i]` (C) `pos[1][i]` (D) `pos[i][1]`