普及组模拟 2-（CSP-J)入门级模拟试卷(二)

(CSP-J)入门级 C++语言试题 模拟卷 - 7

考生注意事项：

1．全部试题答案均要求写在答卷纸上，写在试卷纸上一律无效。

2．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、单选题（共 15 题,每题 2 分,共计 30 分;每题有且仅有一个正确答案

1．有 2N 个数，需要求出最大值和最小值，你需要的最少比较次数是（ ）。

A. 2N+1　　B. 2N+2　　C. 3N-2　　D. 4N-3

2.一片容量为 8GB 的 SD 卡能存储大约(　)张大小为 2MB 的数码照片

A. 1600　　B. 2000　　C. 4000　　D. 16000

3.二进制数 0.1 与十六进制数（）值相等。

A. 0.8　　B. 0.5　　C. 0.1　　D. 0.4

4．如果根结点的深度记为 1，则一棵恰有 2011 个结点的二叉树的深度最少是（ ）。

A. 10　　B. 11　　C. 12　　D. 13

5.设栈 S 的初始状态为空，元素 a，b，c，d，e，f 依次入栈 S，出栈的序列为 b，d，f，e，c，a，则栈 S 的容量至少应该是（ ）。

A. 6　　B. 5　　C. 4　　D. 3

6. 体育课的铃声响了，同学们都陆续地奔向操场，按老师的要求从高到矮站成一排。每个同学按顺序来到操场时，都从排尾走到排头，找到第一个比自己高的同学，并站在他的后面。这种站队的方法类似于（ ）算法。

A. 快速排序　　B. 插入排序　　C. 冒泡排序　　D. 选择排序

7.整形变量 a 和 b,a 的值为 5，执行语句 b=++a 后,b 的值为( )

A. 5　　B. 6　　C. 0　　D. 1

8.n 是一个三位数,那 n 的十位数为()。

A. (n%10)/10　　B. n-n%10　　C. (n/100)%100　　D. (n%100)/10

9.读入一个正整数 a.如果 a 为偶数在屏幕上输出“yes”,如果 a 为奇数在屏幕上输出“no”。为实现该功能程序①处应该填写()。

A. a&1　　B. a|1　　C. !a　　D. a-1

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int a;
5      cin>>a;
6      if(①) cout<<"no";
7      else cout<<"yes";
8  }
```

10. 若二叉树的中序遍历为 ABCDE,则其可能的前序遍历有(　　)种。

A. 14　　B.42　　C. 24　　D.48

11. 若三个整型变量 a,b,mod,满足 a%mod + b%mod == (a+b)%mod,则可得出结论(　　)

A. a 和 b 均小于 mod　　B.任意正整数 a、b、mod 都满足如上表达式

C. a 和 b 均大于 mod　　D.以上都不对

12. 整数数字(-3)^(-5)的值为(　)。（^表示逻辑异或运算）

A. 6　　B.-6　　C.-7　　D.-1

13. 对 13 个有序元素进行二分查找，至多需要几次能够确定所查找的元素是否存在。

A. 12　　B. 5　　C. 4　　D. 3

14.已知某入栈顺序 ABCDEF,第 1 个出栈元素为 C，则栈的出栈序列存在(　　)种。

A. 28　　B. 24　　C. 36　　D. 20

15.以下程序模块执行后输出结果为(　　)。

```
int x = 5;
do{
    cout << x--;
}while(x);
```

A. 54321　　B.4321　　C.43210　　D.无限循环输出

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“X”;除特殊说明外,判断题 1.5 分,选择题 4 分,共计 40 分)

1.

```
#include<bits/stdc++.h>
using namespace std;
int n, m, ans, p[101], e[101];
int pow1(int x, int w) {
    int res = 1;
    for (int i = 1; i <= w; i++) {
        res = res * x;
    }
    return res;
}
int main() {
    cin >> n;
    int i = 2;
    while (n != 1) {
        if (n % i == 0) {
```

```
16              m++;
17              p[m] = i;
18              e[m] = 0;
19              while (n % i == 0) {
20                  e[m]++;
21                  n = n / i;
22              }
23          }
24          i++;
25      }
26      ans = 1;
27      for (int i = 1; i <= m; i ++) {
28          ans = ans * (p[i] - 1) * pow1(p[i], e[i] - 1);
29      }
30      cout << ans << endl;
31  }
```

规定输入的 n 为 int 范围内的正整数。

判断题

1)输出的值一定比输入的值小。(　　)

2)若输入的 n 的值等于 a^b(a, b 均为大于等于 2 的正整数)，则第 14 行循环执行完毕后 m 的值为 1，且 p[1]的值为 a。(　　)

3)存在合法输入使得 n 的值为素数,输出的结果也是素数。(　　)

4)若输入的 n 为素数,程序执行到第 26 行时,i 的值等于一开始输入的 n。(　　)

选择题

5)若输入 19800，则输出(　)。

A.3600　　B.4800　　C.19800　　D.1584000

6)若输出 2400,则输入的 n 可能为 (　　)

A.3200　　B.6400　　C.7200　　D. 9000

2.

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int maxn = 50;
4   void getnext(char str[]) {
5       int len = strlen(str), i, j, k, temp;
6       k = len - 2;
7       while (k - 1 >= 0 && str[k] > str[k + 1]) k--;
8       i = k + 1;
9       while (i < len && str[i] > str[k]) i++;
10      temp = str[k];
11      str[k] = str[i - 1];
12      str[i - 1] = temp;
13      for (i = len - 1; i > k; i--)
14          for (j = k + 1; j < i; j++)
15              if (str[j] > str[j + 1]) {
```

```
16                 temp = str[j];
17                 str[j] = str[j + 1];
18                 str[j + 1] = temp;
19             }
20     return;
21 }
22 int main() {
23     char a[maxn];
24     int n;
25     cin >> a >> n;
26     while (n > 0) {
27         getnext(a);
28         n--;
29     }
30     cout << a << endl;
31     return 0;
32 }
```

判断题

1)若输入的字符串 a 是升序的，那么无论 n 为多少,第 13 行的循环都不会执行(执行的条件不满足)。(　　)

2)若输入的 n 等于 2,对于任意合法字符串输入，第 27 行指令第一次执行完和第二次执行完的字符串最多只有 2 个字符位置不同。(　　)

3)程序执行完第 7 行时,第 k+1 个字符到第 len-1 个字符的值是严格递减。(　　)

4)程序每次执行完第 12 行时，第 k+1 个字符到第 len-1 个字符的值是不严格递减的。(　　)

选择题

5)若输入的字符串有 x 个字符并且是严格降序的,输入的 n 等于 1，则第 16~18 行会执行()次。

A.(x-1)(x-2)/2　　B.x(x-1)/2　　C.x(x+1)/2　　D.0

6）若输入的字符串有 x 个字符并且都相同，输入的 n 等于 1，则第 16~18 行会执行()次。

A.(x-1)(x-2)/2　　B.x(x-1)/2　　C.x(x+1)/2　　D.0

3.

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      int n, p, s, i, j, t;
6      cin >> n >> p;
7      s = 0;
8      t = 1;
9      for (i = 1; i <= n; i++) {
10         t = t * p;
11         for (j = 1; j <= i; j++)
12             s=s+t;
13     }
14     cout<<s<<endl;
```

```
    return 0;
}
```

保证输入值 n 和 p 为正整数，且不考虑数据溢出的影响。

判断题

1）程序运行到第 14 行时,s 和 t 的最大公因数一定大于等于 p。( )

2）程序运行到第 14 行,一定满足 t≥s/2。()

选择题

3)(2 分)程序运行到第 14 行时,t 的值为( )。

A.np B.n^p C.p^n D.p

4)(2 分)当 p=1 时,输出为( )。

A.(n+2)(n+1)/2 B.(n+1)n/2 C.n(n-1)/2 D.n

5)(2 分)若输出为 18555，输入的 n 和 p 是( )。

A.5 5 B.5 6 C.4 7 D. 4 8

6)(3 分)当 n=11, p=2 时,输出为()。

A.2048 B.4096 C.4095 D.40962

三、完善程序(单选题,每小题 3 分,共计 30 分)

1.输入一个正整数 n(2≤n≤10^6)，以及 n 个不同的正整数(范围在 1≤a[i]≤n)。这个数可以理解为 n 的一种排列,假设(1,2,3,4,…,n)是第 1 个排列,(n, n-1,…,1)是最后一个排列,根据这 n 个数组成的排列，输出下一个排列,每一个数后输出一个空格;若这 n 个数已经是最后一个排列，输出“No NextPermutation”。

输入:

5

12543

输出:

13245

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
ll n, num, mid, t;
ll a[10000001], b[1000001];
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    int i = n;
    while (i > 1) {
        if (①){
            num++;
            b[num] = a[i];
        }
        else{
            num++;
```

```
17              b[num] = a[i];
18              num++;
19              ②;
20              mid = i - 2;
21              break;
22          }
23          i--;
24      }
25      if (i == 1)③;
26      else{
27          for (i = 1; i <= num - 1; i++)
28              if (④){
29              t = b[i];
30              b[i] = b[num];
31              b[num] = t;
32              break;
33          }
34          for (i = 1; i <= mid; i++)cout << a[i] << " ";
35          cout << b[num] << " ";
36          for (i = 1; i <= num - 1; i++) ⑤;
37          cout << endl;
38      }
39  }
```

1）①处应填( )。

A. a[i]<a[i-1]　　B. a[i+1]<a[i]　　C.a[i]>a[i-1]　　D.a[i+1]>a[i]

2）②处应填( )。

A.b[num]=a[i-2]　　B.b[num+1]=a[i-1]　　C.b[num+1]=a[i]　　D.b[num]=a[i-1]

3）③处应填( )。

A.num++　　B. mid=0

C. mid--　　D.cout<<"No Next Permuation"

4）④处应填()。

A.b[i]>b[mid]　　B.b[i]<b[mid]

C.b[i]<b[num]　　D.b[i]>b[num]

5）⑤处应填( )。

A.cout<<b[i+1]<<" "　　C. cout<<b[num-i]<<" "

B.cout<<b[i-1]<<" "　　D.cout<<b[i]<<" "

2.(国王放置)在 n x m 的棋盘上放置 k 个国王，要求 k 个国王互相不攻击,有多少种不同的放置方法。假设国王放置在第(x,y)格，国王的攻击区域是(x-1,y-1),(x-1,y),(x-1,y+1),(x,y-1).(x,y+1),(x+1,y-1),(x+1,y)(x+1,y+1)。读入三个数 n, m, k,输出答案。题目利用回溯法求解棋盘行标号为 0～n-1,列标号为 0～m-1。

试补全程序。

```
#include <cstdio>
#include <cstring>
int n, m, k, ans;
int hash[5][5];
void work(int x, int y, int tot) {
    int i, j;
    if (tot == k) {
        ans++;
        return;
    }
    do {
        while (hash[x][y]) {
            y++;
            if (y == m) {
                x++;
                y = ①;
            }
            if (x == n)
                return;
        }
        for (i = x - 1; i <= x + 1; i++)
            if (i >= 0 && i < n)
                for (j = y - 1; j <= y + 1; j++)
                    if (j >= 0 && j < m)
                        ②;
        ③
        for (i = x - 1; i <= x + 1; i++)
            if (i >= 0 && i < n)
                for (j = y - 1; j <= y + 1; j++)
                    if (j >= 0 && j < m)
                        hash[i][j]--;
        ④;
        if (y == m) {
            x++;
            y = 0;
        }
        if (x == n)
            return;
    } while (1);
}
int main( ){
    scanf("%d%d%d", &n, &m,&k);
    ans = 0;
    memset(hash,0, sizeof(hash));
```

```
45      ⑤;
46      printf("%d\n", ans);
47      return 0;
48  }
```

1）①处应填()。

A.0　　B.1　　C.x　　D.x+1

2）②处应填()。

A.hash[i][j]=1　　B.hash[i][j]++　　C. hash[i][j]=0　　D.hash[i][j]--

3）③处应填( )。

A. work(x, y+1, tot+1)　　B. y--　　C.y++　　D.work(x, y, tot+1)

4）④处应填()

A.y=0　　B. y=1　　C. y--　　D. y++

5）⑤处应填( )。

A.work(n-1,m-1,0)　　C. work(0,0,0)

B.work(n,m,0)　　D.work(1,1,0)