GESP 四级模拟卷(1)

一、选择题

1. 以下说法中正确的是(　　B　　)

A、main 函数和其他函数间可相互调用

B、main 函数可以调用其他函数，但其他函数不能调用 main 函数

C、因为 main 函数可不带参数，所以其后的参数小括号能省略

D、根据情况可以不写 main 函数

2. C++语言规定，函数返回值的类型是(　　C　　)

A、由 return 语句中表达式的类型所决定

B、由调用该函数的主调函数所具有的类型决定

C、由定义该函数时所指定的函数类型决定

D、有系统随即决定

3. C 语言规定，简单变量做实参时，它和对应的形参之间的数据传递方式是（　　B）。

A.地址传递

B.值传递

C.由实参传给形参，再由形参传给实参

D.由用户指定传递方式

4. 以下正确的函数声明形式是（ D ）。

A.int fun(int x,int y)

B.int fun(int x;int y)

C.int fun(int x, y);

D.int fun(int x,int y);

5. 若有数组名作为函数调用的实参，传递给形参的是（A ）。

A.数组的首地址

B.数组第一个元素的值

C.数组中全部元素的值

D.数组元素的个数

6. 执行如下程序段，打印输出的内容是(　A　)：

```
void fun (int c, int *d) {
    c++;
    (*d)++;
}
int main ( ){
    int a=1, b=2;
    fun(a, &b);
    printf("%d, %d", a, b);
    return 0;
}
```

A.1, 3　　B.2, 1　　C.1, 2　　D.2, 2

7. 若变量已正确定义并且指针 p 已经指向某个变量 x，则(*p)++相当于(　B　)。

A. p++ B.x++ C.*(p++) D.&x++

8. 设变量定义为 int a[2]={1,3}, *p=&a[0]+1;，则*p 的值是(B)。

A.2 B.3 C.4 D.&a[0]+1

9. 下面程序段执行后的输出结果是（C ）。

```
char a[] = "language", *p;
p = a;
while ( *p != 'u' )
{
    printf( "%c", *p - 32 ); p++;
}
```

A.LANGUAGE B.language

C.LANG D.langUAGE

10. 数组定义为 int a[3][2]={1, 2, 3, 4, 5, 6}，数组元素(B)的值为6。

A.a[3][2] B.a[2][1] C.a[1][2] D.a[2][3]

11.

```
int main( )
 {
    char ch[2][5]={"6934","8254"},*p[2];
    int i,j,s=0;
    for(i=0;i<2;i + +)
     p[i]=ch[i];
    for(i=0;i<2;i + +)
     for(j=0;p[i][j]>='0'&&p[i][j]<='9';j+=2)
       s=10*s+p[i][j]-'0';
    printf("%d\n",s);
    return 0;
 }
```

以上程序运行结果是：(A)

A.6385 B.69825 C.63825 D.693825

12. 下列排序算法中，哪种算法可能出现：在最后一趟开始之前，所有的元素都不在其最终的位置上？（设待排元素个数N>2）(B)

A.冒泡排序 B.插入排序 C.选择排序 D.快速排序

13. 对于 7 个数进行冒泡排序，需要进行的比较次数为：(C)

A.7 B.14 C.21 D.49

14. 对一组数据{ 2，12，16，88，5，10 }进行排序，若前三趟排序结果如下：

第一趟排序结果：2，12，16，5，10，88

第二趟排序结果：2，12，5，10，16，88
第三趟排序结果：2，5，10，12，16，88
则采用的排序方法可能是（ A ）：

A. 冒泡排序　　B. 希尔排序　　C. 插入排序　　D. 选择排序

15. 一个问题可用动态规划法或贪心法求解的关键特征是问题的（ C ）。

A. 贪心选择性质　　B. 重叠子问题
C. 最优子结构性质　　D. 定义最优解

二、判断题

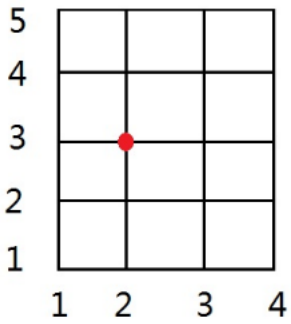
1. 函数的实参传递到形参有两种方式：值传递和地址传递。（T）
2. 函数形参的存储单元是动态分配的。(T)
3.局部变量如果没有指定初值，则其初值不确定。(T)
4. 按照 C++语言的规定，在参数传递过程中，既可以将实参的值传递给形参，也可以将形参的值传递给实参，这种参数传递是双向的。（F）
5. 被调函数中必须有 return 语句，通过它可以带回一个返回值。(F)
6. 全局变量只能定义在程序的最前面，即第一个函数的前面。(F)
7. 全局变量不可以和函数内的局部变量同名。(F)
8. 对于已正确定义的二维数组 a, *(a[i]+j)与 a[i][j]的含义相同。(T)
9.二维数组的元素在内存中按行/列方式存放，即先存放第 0 行的元素，再存放第 1 行的元素……其中每一行的元素再按照列的顺序存放。（T）
10.数组定义后，数组名表示该数组所分配连续内存空间中第一个单元的地址，即首地址，是一个常量，不能被修改。(T)
11. 基于比较的排序算法中，只要算法的最坏时间复杂度或者平均时间复杂度达到了次平方级 O(N * logN)，则该排序算法一定是不稳定的。(F)
12. 排序的稳定性是指排序算法中的比较次数保持不变，且算法能够终止。(F)
13. 对 N 个不同的数据采用冒泡排序进行从大到小的排序，当元素基本有序时交换元素次数肯定最多。(F)
14. 执行语句 int *p = 1000;后，指针变量 p 指向地址为 1000 的变量。(F)
15. c 语言源程序是文本文件，目标文件和可执行文件是二进制文件。(T)

三、编程题

1. 桌子上有一个 m 行 n 列的方格矩阵，将每个方格用坐标表示，行坐标从下到上依次递增，列坐标从左至右依次递增，左下角方格的坐标为(1,1)，则右上角方格的坐标为(m,n)。

小明是个调皮的孩子，一天他捉来一只蚂蚁，不小心把蚂蚁的右脚弄伤了，于是蚂蚁只能向上或向右移动。小明把这只蚂蚁放在左下角的方格中，蚂蚁从 左下角的方格中移动到右上角的方格中，每步移动一个方格。蚂蚁始终在方格矩阵内移动，请计算出不同的移动路线的数目。

对于 1 行 1 列的方格矩阵，蚂蚁原地移动，移动路线数为 1；对于 1 行 2 列（或 2 行 1 列）的方格矩阵，蚂蚁只需一次向右（或向上）移动，移动路线数也为 1……对于一个 2 行 3 列的方格矩阵，如下图所示：



蚂蚁走到（2,3）共有 3 种移动路线：

路线 1：(1,1) → (1,2) → (1,3) → (2,3)

路线 2：(1,1) → (1,2) → (2,2) → (2,3)

路线 3：(1,1) → (2,1) → (2,2) → (2,3)

输入格式

输入只有一行，包括两个整数 m 和 n（0 < m+n ≤ 20），代表方格矩阵的行数和列数，m、n 之间用空格隔开。

输出格式

输出只有一行，为蚂蚁爬到（m,n）时不同的移动路线的数目。

输入样例

2 3

输出样例

3

2. [数值排序]

输入一串数字，把这串数字中的‘0’都看成空格，那么就得到一行用空格分割的若干非负整数（如果有三个或三个以上连续的‘0’，则第 1 个‘0’看成一个空格，第 2 个‘0’看成数值 0，后面连续的 0 看成一个空格）。

你的任务是对这些分割得到的整数依从大到小的顺序排序。分割出来的整数 x<=1018

输入格式

没个测试数据为一串数字（数字之间没有空格，最前面的数字不为 0）。测试数据保证：分割后得到的每个非负整数不会大于 10000000000；分割后最多有 10000 个非负整数；最后一个数字不可能是 0。

请用字符处理。

输出格式

输出一行，是分割得到的整数排序的结果，相邻的两个整数之间用一个空格分开。

输入 2：

11105430002398001298009000098765430001

输出：9876543 2398 1298 543 111 9 1 0 0 0

输入样例 复制

4500051231232050775

输出样例 复制

51231232 775 45 5 0

1.

```
#include <iostream>
using namespace std;
int a[25][25],f[25][25];
int main()
{
   int n,m;
   cin>>m>>n;
   for(int i=m;i>=1;i--)
   {
      for(int j=1;j<=n;j++)
      {
         f[i][j]=f[i][j-1]+f[i+1][j];
         f[m][1]=1;
      }
   }
   cout<<f[1][n];
   return 0;
}
```

2.

```
#include <bits/stdc++.h>
using namespace std;

long long a[10001];

int cmp(long long a, long long b){
      return a>b;
}
int main()
{//343000435300123
//4353 343 0 123
      int k,len;
      char s[100000];
      cin>>s;
      k=0;
      len=strlen(s);
      for (int i=0;i<len;i++){
            if (s[i]>'0')
                  a[k]=a[k]*10+s[i]-'0';
            else
            {
                  k++;
                  if (s[i+1]=='0'&&s[i+2]=='0') k++;
```

```
                while (s[i+1]=='0') i++;
            }
        }
        k=k+1;//元素的个数
        sort(a,a+k,cmp);
        for (int i=0;i<k;i++) printf("%lld ",a[i]);

        return 0;
}
```