

每日一题day16_6月6日测评结果

考生信息



张博翔

考号：1675 | 学校：陕西科技大学 | 邮箱：1761607418@qq.com | 职位：43班 |

参考区域: 陕西省西安市 (111.114.0.2) | 做题用时：06:52:02(2019-06-06 11:41:57 - 18:34:02)

考生成绩



知识点技能图谱



历史笔试记录

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	笔试时间
1	每日一题day1_5月20日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-19 16:08:49
2	每日一题day02_5月21日	51.0%	60.0/100	单选:30.0分 编程:30.0分	否	2019-05-20 17:40:56
3	每日一题day03_5月22日	4.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-21 12:59:15
4	每日一题day04_5月23日	8.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-22 10:40:18
5	每日一题day05_5月24日	29.0%	75.0/100	单选:25.0分 编程:50.0分	否	2019-05-22 20:39:16

序号	试卷名称	排名	总得分	得分详情	作弊嫌疑	笔试时间
6	每日一题day06_5月25日	10.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-24 10:41:21
7	每日一题day07_5月27日	6.0%	95.0/100	单选:45.0分 编程:50.0分	否	2019-05-26 15:09:41
8	每日一题day08_5月28日	18.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-27 14:06:31
9	每日一题day09_5月29日	13.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-28 15:11:53
10	每日一题day10_5月30日	6.0%	90.0/100	单选:40.0分 编程:50.0分	否	2019-05-29 11:32:20
11	每日一题day11_5月31日	14.0%	80.0/100	单选:30.0分 编程:50.0分	否	2019-05-30 11:22:09
12	每日一题day12_6月1日	31.0%	75.0/100	单选:25.0分 编程:50.0分	否	2019-05-31 10:33:31
13	每日一题day13_6月3日	9.0%	85.0/100	单选:35.0分 编程:50.0分	否	2019-06-02 15:20:12
14	每日一题day14_6月4日	41.0%	50.0/100	单选:25.0分 编程:25.0分	否	2019-06-03 11:53:13
15	每日一题day15_6月5日	6.0%	85.0/100	单选:35.0分 编程:50.0分	否	2019-06-04 12:02:48

编码能力



1 [平均分1.7分 | 38人正确/113人做题 | 用时 : <1分] 得分 : 0.0 / 5.0

```
void swap_int(int *a,int *b){
    *a=*a+*b;
    *b=*a-*b;
    *a=*a-*b;
}
```

- 以下说法正确的是：
- A 结果不正确，因为会溢出，用位与的方式就没问题
 - B 结果正确，即使会溢出
 - C 结果正确，不会溢出
 - D 其他选项都不对

他的回答： A (错误)

正确答案： B

2 [平均分2.8分 | 61人正确/110人做题 | 用时 : <1分] 得分 : 5.0 / 5.0

- 若有定义int (*pt) [3]；则下列说法正确的是：
- A 定义了基类型为int的三个指针变量
 - B 定义了基类型为int的具有三个元素的指针数组pt
 - C 定义了一个名为*pt、具有三个元素的整型数组
 - D 定义了一个名为pt的指针变量，它可以指向每行有三个整数元素的二维数组

他的回答： D (正确)

正确答案：D

3 [平均分2.3分 | 51人正确/112人做题 | 用时：<1分] 得分：5.0 / 5.0

对于下面的说法，正确的是_____。

- A 对于 struct X { short s; int i; char c; } , sizeof(X) 的值等于 sizeof(s) + sizeof(i) + sizeof(c)
- B 对于某个double变量 a , 可以使用 a == 0.0 来判断其是否为零
- C 初始化方式 char a[14] = "Hello, world!"; 和初始化方式 char a[14]; a = "Hello, world!"; 的效果相同
- D 在gcc编译器下 , 对于 int i = 3; printf("%d %d", ++i, ++i) , 运行输出为 : 4 5
- E 选项A、B、C、D中至少有两个是正确的
- F 以上选项均不正确

他的回答：F (正确)

正确答案：F

4 [平均分2.1分 | 46人正确/112人做题 | 用时：17分] 得分：5.0 / 5.0

下列代码的输出是？（注：print已经声明过）

```
main(){
    char str[]="Geneius";
    print(str);
}
print(char *s){
    if(*s){
        print(++s);
        printf("%c",*s);
    }
}
```

- A suiene
- B neius
- C run-time error
- D suieneG

他的回答：A (正确)

正确答案：A

5 [平均分3.5分 | 74人正确/105人做题 | 用时：9分] 得分：5.0 / 5.0

写出下面程序的输出结果

```
class A
{
public:
    void FuncA()
    {
        printf( "FuncA called\n" );
    }
    virtual void FuncB()
    {
        printf( "FuncB called\n" );
    }
};
class B : public A
{
public:
```

```

void FuncA()
{
    A::FuncA();
    printf( "FuncAB called\n" );
}
virtual void FuncB()
{
    printf( "FuncBB called\n" );
}
};
void main( void )
{
    B b;
    A *pa;
    pa = &b;
    A *pa2 = new A;
    pa->FuncA(); ( 3 )
    pa->FuncB(); ( 4 )
    pa2->FuncA(); ( 5 )
    pa2->FuncB();
    delete pa2;
}

```

- A FuncA called FuncB called FuncA called FuncB called
 B FuncA called FuncBB called FuncA called FuncB called
 C FuncA called FuncBB called FuncAB called FuncBB called
 D FuncAB called FuncBB called FuncA called FuncB called

他的回答： B (正确)

正确答案： B

6 [平均分2.7分 | 58人正确/109人做题 | 用时：7分] 得分：5.0 / 5.0

以下程序输出是_____。

```

#include <iostream>
using namespace std;
int main(void)
{
    const int a = 10;
    int * p = (int *)(&a);
    *p = 20;
    cout<<"a = "<<a<<" , *p = "<<*p<<endl;
    return 0;
}

```

- A 编译阶段报错运行阶段报错
 B a = 10, *p = 10
 C a = 20, *p = 20
 D a = 10, *p = 20
 E a = 20, *p = 10

他的回答： D (正确)

正确答案： D

7 [平均分2.0分 | 43人正确/110人做题 | 用时：<1分] 得分：0.0 / 5.0

以下关于STL的描述中，_____是错的。

- A STL容器是线程不安全的
- B 当容量不够时，vector内部内存扩展方式是翻倍
- C std::sort是稳定排序
- D std::bitset不是一个STL容器
- E std::stack默认是用deque实现的
- F std::string中可以存储多个'\0'字符

他的回答： B (错误)

正确答案： C

参考答案：

要是能在这里留下这道题的解题思路，就再好不过啦

8 [平均分1.9分 | 43人正确/111人做题 | 用时：17分] 得分：0.0 / 5.0

以下代码共调用多少次拷贝构造函数：

```
Widget f(Widget u)
{
    Widget v(u);
    Widget w=v;
    return w;
}
main(){
    Widget x;
    Widget y=f(f(x));
}
```

- A 1
- B 3
- C 5
- D 7

他的回答： C (错误)

正确答案： D

9 [平均分1.2分 | 27人正确/109人做题 | 用时：32分] 得分：5.0 / 5.0

以下代码有什么问题？

```
struct Test
{
    Test( int ) {}
    Test() {}
    void fun() {}
};
void main( void )
{
    Test a(1);
    a.fun();
    Test b();
    b.fun();
}
```

- A b.fun () 会出错
- B Test结构的定义中应该加上public修饰符，这样才能main函数中调用改类的方法
- C Test(int){} 应该改成Test(int a){}
- D 以上说法都不正确

他的回答：A (正确)

正确答案：A

10 [平均分2.4分 | 50人正确/105人做题 | 用时：<1分 | 得分：0.0 / 5.0]

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

class Base
{
public:
    virtual int foo(int x)
    {
        return x * 10;
    }

    int foo(char x[14])
    {
        return sizeof(x) + 10;
    }
};

class Derived: public Base
{
    int foo(int x)
    {
        return x * 20;
    }

    virtual int foo(char x[10])
    {
        return sizeof(x) + 20;
    }
};

int main()
{
    Derived stDerived;
    Base *pstBase = &stDerived;

    char x[10];
    printf("%d\n", pstBase->foo(100) + pstBase->foo(x));

    return 0;
}
```

在32位环境下，以上程序的输出结果是？

- A 2000
- B 2004
- C 2014
- D 2024

他的回答：B (错误)

正确答案：C

11 [平均分22.5分 | 92人正确/103人做题 | 提交：1次 | 得分：25.0 / 25.0]

标题：iNOC产品部-完全数计算 | 时间限制：1秒 | 内存限制：32768K | 语言限制：不限

【iNOC产品部--完全数计算】

完全数（ Perfect number ），又称完美数或完备数，是一些特殊的自然数。
它所有的真因子（即除了自身以外的约数）的和（即因子函数），恰好等于它本身。
例如：28，它有约数1、2、4、7、14、28，除去它本身28外，其余5个数相加，1+2+4+7+14=28。
给定函数count(int n),用于计算n以内(含n)完全数的个数。计算范围, 0 < n <= 500000
返回n以内完全数的个数。 异常情况返回-1

```
/**
 *
 * 完全数（ Perfect number ），又称完美数或完备数，是一些特殊的自然数。
 * 它所有的真因子（即除了自身以外的约数）的和（即因子函数），恰好等于它本身。
 * 例如：28，它有约数1、2、4、7、14、28，除去它本身28外，其余5个数相加，1+2+4+7+14=28。
 *
 * 给定函数count(int n),用于计算n以内(含n)完全数的个数
 * @param n 计算范围, 0 < n <= 500000
 * @return n 以内完全数的个数, 异常情况返回-1
 */
public static int count( int n)
```

输入描述：

输入一个数字

输出描述：

输出完全数的个数

示例1：

输入

1000

输出

3

代码片段

功能实现			代码提交统计			代码执行统计
总通过率	TA的 100%	平均 89%	使用语言	TA的 C++	平均	答案正确：1
基本测试用例通过率	6/6 (100%)	89%	做题用时	00:14:37	00:31:14	
边缘测试用例通过率	4/4 (100%)	89%	提交次数	1	4	
代码效率			代码规范及可读性			
运行时间	TA的 126ms	参考 1s	代码规范得分 5.0			
占用内存	480K	32768K				

他的代码：

做题用时: 14 分钟 语言：C++ 运行时间：126ms 占用内存：480K 程序状态：答案正确

```
#include <iostream>
#include <vector>
using namespace std;
bool isPerfect(int num)
{
    vector<int> primes;
    for(int i = 1; i < num; i++)
    {
        if(num % i == 0)
        {
```

```

        primes.push_back(i);
    }
}
int sum = 0;
for(int i = 0; i < primes.size(); i++)
{
    sum += primes[i];
}
if(sum == num)
{
    return true;
}
return false;
}
int main()
{
    int num = 0;
    while(cin >> num)
    {
        int count = 0;
        for(int i = 1; i <= num; i++)
        {
            if(isPerfect(i))
            {
                count++;
            }
        }
        cout << count << endl;
    }
}

```

12 [平均分18.4分 | 35人正确/61人做题 | 提交: 7 次 | 得分: 25.0 / 25.0

标题: 扑克牌大小 | 时间限制: 1秒 | 内存限制: 32768K | 语言限制: 不限

【扑克牌大小】

扑克牌游戏大家应该都比较熟悉了，一副牌由54张组成，含3~A、2各4张，小王1张，大王1张。牌面从小到大用如下字符和字符串表示（其中，小写joker表示小王，大写JOKER表示大王）：

3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER

输入两手牌，两手牌之间用“-”连接，每手牌的每张牌以空格分隔，“-”两边没有空格，如：4 4 4 4-joker JOKER。

请比较两手牌大小，输出较大的牌，如果不存在比较关系则输出ERROR。

基本规则：

- （1）输入每手牌可能是个子、对子、顺子（连续5张）、三个、炸弹（四个）和对王中的一种，不存在其他情况，由输入保证两手牌都是合法的，顺子已经从小到大排列；
- （2）除了炸弹和对王可以和所有牌比较之外，**其他类型的牌只能跟相同类型的存在比较关系（如，对子跟对子比较，三个跟三个比较）**，不考虑拆牌情况（如：将对子拆分成个子）；
- （3）大小规则跟大家平时了解的常见规则相同，个子、对子、三个比较牌面大小；顺子比较最小牌大小；炸弹大于前面所有的牌，炸弹之间比较牌面大小；对王是最大的牌；
- （4）输入的两手牌不会出现相等的情况。

输入描述：

输入两手牌，两手牌之间用“-”连接，每手牌的每张牌以空格分隔，“-”两边没有空格，如 4 4 4 4-joker JOKER。

输出描述：

输出两手牌中较大的那手，不含连接符，扑克牌顺序不变，仍以空格隔开；如果不存在比较关系则输出ERROR。

示例1：

输入

4 4 4 4-joker JOKER

输出

joker JOKER

代码片段

功能实现			代码提交统计			代码执行统计	
总通过率	TA的 100%	平均 73%	使用语言	C++		答案错误	: 5
基本测试用例通过率	6/6 (100%)	74%	做题用时	05:00:25	01:26:59	答案正确	: 2
边缘测试用例通过率	4/4 (100%)	72%	提交次数	7	4		
代码效率					代码规范及可读性		
运行时间	TA的 4ms	参考 1s				代码规范得分	5.0
占用内存	468K	32768K					

他的代码：

做题用时: 300 分钟 语言：C++ 运行时间：4ms 占用内存：468K 程序状态：答案正确

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int NumOfPoker(string str)
{
    int count = 1;
    for(size_t i = 0; i < str.size(); i++)
    {
        if(str[i] == ' ')
        {
            count++;
        }
    }
    return count;
}
int PriceOfPoker(string str)
{
    int res = 0;
    /*
    if(str == "joker")
    {
        res = 14;
    }
    if(str == "JOKER")
    {
        res = 15;
        return res;
    }
    */
    if(str[0] == '3')
    {
        res = 1;
    }
    if(str[0] == '4')
    {
        res = 2;
    }
}
```

```
}
if(str[0] == '5')
{
    res = 3;
}
if(str[0] == '6')
{
    res = 4;
}
if(str[0] == '7')
{
    res = 5;
}
if(str[0] == '8')
{
    res = 6;
}
if(str[0] == '9')
{
    res = 7;
}
if(str[0] == '1')
{
    res = 8;
}
if(str[0] == 'J')
{
    res = 9;
}
if(str[0] == 'Q')
{
    res = 10;
}
if(str[0] == 'K')
{
    res = 11;
}
if(str[0] == 'A')
{
    res = 12;
}
if(str[0] == '2')
{
    res = 13;
}
return res;
}
int main()
{
    string str;
    while(getline(cin, str))
    {
        string player1 = str.substr(0, str.find('-'));//玩家一
        string player2 = str.substr(str.find('-') + 1);//玩家二
        //判断胜负
        if(player1 == "joker JOKER")
        {
            cout << player1 << endl;
        }
        else if(player2 == "joker JOKER")
        {

```

```
        cout << player2 << endl;
    }
    else if(NumOfPoker(player1) == NumOfPoker(player2))
    {
        if(PriceOfPoker(player1) > PriceOfPoker(player2))
        {
            cout << player1 << endl;
        }
        else
        {
            cout << player2 << endl;
        }
    }
    else if(NumOfPoker(player1) == 4)
    {
        cout << player1 << endl;
    }
    else if(NumOfPoker(player2) == 4)
    {
        cout << player2 << endl;
    }
    else
    {
        cout << "ERROR" << endl;
    }
}
}
```