day\_1

1.在（）情况下适宜采用 inline 定义内联函数

A 函数体含有循环语句 B 函数体含有递归语句 C 函数代码少、频繁调用 D 函数代码多，不常调用

错选 D------正确答案C

解析:内联函数的使用场景：（1）函数本身内容比较少，代码比较短，函数功能简单

                                                 （2）函数被调用的频繁，比如循环中的函数

         内联函数的利与弊：利：避免了指令来回跳转；弊：代码被复制了多次，增加了代码量，占用更多空间

         不适用的场景：函数代码多，功能复杂，体积庞大；递归函数

2.在 c++ 语言中，对函数参数默认值描述正确的是（）

 A 函数参数的默认值只能设定一个

 B 一个函数的参数若有多个，则参数默认值的设定可以不连续

 C 函数参数必须设定默认值

 D 在设定了参数的默认值后，该参数后面定义的所有参数都必须设定默认值

解析：对于A选项来说，函数参数的默认值可以设置多个；对于B选项，函数参数若有多个时，设置规则要么是全缺省，要么是半缺省。对于带参数的函数，设置默认值的方向必须是从左到右。对于C选项，函数参数在不设置默认参数时，默认初始化为0；对与选项D：在设定了参数的默认值以后，后面的参数必须设定默认值，否则编译器无法判断在调用的时候是否有参数遗漏。

【组队竞赛】牛牛举办了一次编程比赛,参加比赛的有3\*n个选手,每个选手都有一个水平值a\_i.现在要将这些选手进行组队,一共组成n个队伍,即每个队伍3人.牛牛发现 队伍的水平值等于该队伍队员中第二高水平值。

 例如: 一个队伍三个队员的水平值分别是3,3,3.那么队伍的水平值是3

           一个队伍三个队员的水平值分别是3,2,3.那么队伍的水平值是3

           一个队伍三个队员的水平值分别是1,5,2.那么队伍的水平值是2

           为了让比赛更有看点,牛牛想安排队伍使所有队伍的水平值总和最大。

           如样例所示: 如果牛牛把6个队员划分到两个队伍

           如果方案为: team1:{1,2,5}, team2:{5,5,8}, 这时候水平值总和为7.

           而如果方案为: team1:{2,5,8}, team2:{1,5,5}, 这时候水平值总和为10.

           没有比总和为10更大的方案,所以输出10.

#include <vector>

#include <algorithm>

#include <iostream>

using namespace std;

int main()

{

    int n = 0;

    while (cin >> n)

    {

        vector<int>nums(3 \* n);

        long max = 0;

        for (int i = 0; i <nums.size();++i)

            cin>>nums[i];

        sort(nums.begin(),nums.end());

        for (int i = n;i <= nums.size() - 2;i += 2)

            max += nums[i];

        cout<<max<<endl;

    }

}

【删除公共字符】输入两个字符串，从第一字符串中删除第二个字符串中所有的字符。例如，输入”They are students.”和”aeiou”，则删除之后的第一个字符 串变成”Thy r stdnts.”

//在第一个字符串中删除第二个字符串中出现的所有字母

#include<iostream>

#include <algorithm>

using namespace std;

class Solution

{

public:

    void delect\_str(string &str1,string &str2)

    {

        if (str1.empty() || str2.empty())

            return;

        for (int i = 0;i < str1.size();++i)

        {

            for (int j = 0;j < str2.size();++j)

            {

                int pos = str1.find(str2[j]);

                if (pos == -1)

                    continue;

                str1.erase(pos,1);

            }

        }

    }

};

int main()

{

    std::string str1;

    std::string str2;

    std::getline(std::cin,str1);

    std::getline(std::cin,str2);

    Solution S;

    S.delect\_str(str1,str2);

    cout<<str1<<endl;

}