第十五届stema选拔赛C++中高级试卷（3月10日）

一、选择题（50分）

```
1、(110010)2 + (c3)16的结果是（  ）
A、(240)10
B、(11110101)2
C、(366)8
D、(f6)16
注意：此题目()后面的数代表进制，入(366)8，代表是八进制下的366这个数
```

参考答案：B

知识点考查：进制转换

解析：题目中出现了2、8、10、16四种不同进制的数据，为了方便计算，我们可以统一转化为10进制，(110010)2按权展开得到的十进制数为2+16+32=50，(c3)16按权展开相加为：3×16^0 + c×16^1=195，因此(110010)2 + (c3)16=(245)10

A选项：题目算式的结果为十进制的195，A选项为十进制的**240**，因此**A错**

B选项：**根据二进制按权展开，逐项相加的转化规则，(11110101)2所对应的十进制为1+4+16+32+64+128=245，因此B正确****

C选项：根据八进制按权展开，逐项相加的转化规则，(366)8所对应的十进制为6×8^0+6×8^1+3×8^2=**246**，因此**C错**

D选项：根据十六进制按权展开，逐项相加的转化规则，(f6)16所对应的十进制为6×16^0+f×16^1=246，因此**D错**

```
2、表达式1000/3的结果是（  ）
A、333
B、333.3
C、334
D、333.0
```

参考答案：A

知识点考查：C++中算术运算符/的使用规则

解析：在C++编程中，如果"/"左右两边的数据都为整数类型，比如5/2，因为5和2都是整数类型，那么得到的结果也是整数类型，为2，而不是2.5，如果想让结果为2.5，那么需要把/左右变量的任意一方变为浮点类型或者两边都为浮点类型，比如5.0/2或5/2.0或5.0/2.0。

A选项：**1000为整数类型，3为整数类型，那么得到的结果也为整数类型，即333，而不是333.333……，因此A正确**

B选项：333.3为小数类型，错误，如果要得到333.3，那么代码书写如右：`printf("%.1f",1000.0/3);`，因此**B错**

C选项：四舍五入保留0位小数也是得到333，不会得到334，验证代码如右：`printf("%.0f",1000.0/3);`，因此**C错**

D选项：计算的结果是整数类型，不是小数类型，333.0错误，因此**D错**

```
3、下列选项中，判断a等于1并且b等于1正确的表达式是（　）
A、!((a!=1)&&(b!=1))
B、!((a!=1)||(b!=1))
C、!(a==1)&&(b==1)
D、(a=1)&&(b=1)
```

参考答案：B

知识点考查：逻辑运算符&&、||、!的使用

解析：在C++编程中，&&代表逻辑与，用法为：表达式1&&表达式2，当且仅当表达式1和表达式2同时为真，整个**表达式1&&表达式2**为真；||代表逻辑或，用法为：表达式1||表达式2，只要表达式1或者表达式2其中有一个为真，整个**表达式1&&表达式2**为真；|代表逻辑非，取反，它能够把真变为假，把假变为真，如!true等于false，!0等于1；。

A选项：举个反例，如果a=2,b=1,那么表达式的结果也为真，这是不允许的，当且仅当a和b同时为1，表达式的结果才能为真，其它情况应该为假，因此**A错**

B选项：**!((a!=1)||(b!=1))表达式的意思为只要a或者b其中有一个不等于1，那么表达式的结果为假/不成立，当且仅当a和b同时等于1，整个表达式的结果才为真，因此B正确**

C选项：举个反例，如果a等于1,b等于1，那么代入表达式，!(a1)&&*(b1)等价于!(11)&&(11)，该表达式的结果为false，为假/不成立，这与题目的意思相违背，当a和b同时等于1的时候，表达式的值应该为真才对，因此**C错**

D选项：将a赋值为1，b赋值为1，1为真，那么a&&b等于1&&1，表达式的结果永远为1，因此**D错**

```
4、定义 char a[] = "His name is Jack"，请问 sizeof(a) 的结果是（　）
A、14
B、15
C、16
D、17
```

参考答案：D

知识点考查：sizeof运算符的理解与使用、字符串初始化方式

解析： sizeof是一个关键字，一个运算符，它可以返回一个对象或者某种数据类型产生的变量所占的内存字节数，以字符串""的格式进行数据的初始化需要特别注意：在整个字符串的末尾，会自动补上字符串结束标记字符\0，\0是一个转义字符，因此本题中字符串数组a在内存中所占的字节数为"His name is Jack"的字符长度+\0这一个字符，即16+1=17；代码验证如：`char a[] = "His name is Jack"; cout << sizeof(a);`。

A选项：A选项为14字节，因此**A错**

B选项：B选项为15字节，因此**B错**

C选项：C选项为16字节，因此**C错**

D选项：**D选项有考虑\0的问题，D选项为17个字节，因此D正确**

```
5、定义 int a[] = {5,1,3,8,2,9,0,6}, *p=(a+3)，那么((*p)--+ *p)的值是（  ）
A、3
B、10
C、15
D、16
```

> **参考答案：D**
>
> **知识点考查：指针与数组的关系，指针的概念与应用，自增、自减运算符的使用方式**
>
> **解析1：** 对于小学生来说，指针是一个难点，指针变量主要用来存储地址，所谓的地址，就是内存的编号，举一个简单的例子，以整型变量为例，即int a=5，代表在内存中开辟一片空间，放的数据为5，其实在内存中，这片空间是有编号的，我们称为地址，就像是家里的门牌号一样，指针变量可以用来存储变量空间的编号，比如 int *p = &a，代表p是一个整型指针变量，它能够存储整型变量空间的编号，&a代表获得a变量空间的编号，整个语句代表，获取整型变量a空间的编号，然后把整个编号存储到p指针变量中，这个时候，我们称p指向a空间，`*p就等价于a变量本身，操作*p就是操作a变量`。
>
> ```
> 解析代码示例如下：
> int a = 5;
> int *p = &a;
> //*p等价于a，用*p就相当于在用a这个变量
> *p = 100;
> cout << a;
> ```
>
> **解析2：** a代表数组的首地址，那么a+3这个地址就是a[3]这个元素所在的地址，int *p=(a+3)，那么就意味着p这个整型的指针变量中存储的是整型数组中数组元素a[3]的地址，此时`*p等价于a[3]，使用*p就是使用a[3]这个数据元素本身`，对`((*p)--+ *p)`进行拆解，`(*p)--`就等于a[3]--，整个表达式的返回值还是8，因为--在后面，那么a[3]的值由原来的8变为7，`*p等价于a`，因此`((*p)--+ *p)=8+7=15`。
>
> **A选项**：A选项为3，因此**A错**
>
> **B选项**：B选项为10，因此**B错**
>
> **C选项**：**C选项为15，因此C正确**
>
> **D选项**：D选项为16，因此D错

二、编程题（350分）

`时间限制：1000MS，内存限制：65536KB`

> **（1）题目描述（编程实现）：** 寒假期间小明需要做完n张试卷，但他每天最多能做完m 张，请计算出小明做完n张试卷最少需要多少天?
>
> **输入描述：** 一行输入两个整数n和m（1≤n≤100，1≤m≤10），分别表示要完成的试卷张数，及每天最多能做完的试卷张数，整数之间以一个空格隔开
>
> **输出描述：** 输出一个整数，表示小明最少多少天能做完n张试卷
>
> **样例输入：** 10 3
>
> **样例输出：** 4

```
评分标准
2分：能正确输出第一组数据；
2分：能正确输出第二组数据；
2分：能正确输出第三组数据；
2分：能正确输出第四组数据；
2分：能正确输出第五组数据。
2分：能正确输出第六组数据；
2分：能正确输出第七组数据；
2分：能正确输出第八组数据；
2分：能正确输出第九组数据；
2分：能正确输出第十组数据。
```

> **知识点考查**：**/ 与 % 的使用方式 或者 if 的使用方式**
>
> **思路解析：判断n能否被m整除，如果可以，那么有n张试卷，每天做m张，那么需要做n/m天；如果n不能被m整除，那么就需要再多花一天的时间，即n/m+1天，比如n=10,m=3，n%m!=0，于是最少需要n/m+1=4天才能做完n张试卷**

参考代码1：

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

int n, m;
int main() {
    cin >> n >> m;
    // (n % m != 0) 如果n能被m整除，那么结果为false
    //false为0，即n / m + (n % m != 0) = n/m
    //如果n不能被m整除，那么结果为true
    //true为1，即n / m + (n % m != 0) = n/m + 1
    cout << n / m + (n % m != 0);
    return 0;
}
```

参考代码2：

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

int n, m;
int main() {
    cin >> n >> m;
    if(n % m == 0) cout << n / m;
    else cout << n / m + 1;
    return 0;
}
```

`时间限制：1000MS，内存限制：65536KB`

> **提示信息：**
>
> 回文数：一个自然数从左往右读和从右往左读都一样，则该自然数被称为回文数，例如：121是回文数，123不是回文数。

> **（2）题目描述（编程实现）：**给定两个整数a，b，请统计a到b之间（包含a和b）有多少个包含数字7的回文数。
>
> 例如：a=6，b=80，6到80之间的回文数有6、7、8、 9、11、22、33、44、55、66、77，其中有2个回文数包含7（7和77）。
>
> **输入描述：**一行输入两个整数a和b（1≤a≤b≤100000），整数之间以一个空格隔开
>
> **输出描述：**输出一个整数，表示a到b之间（包含a和b）包含数字7的回文数的个数
>
> **样例输入：**6 80
>
> **样例输出：**2

```
评分标准：
4分：能正确输出第一组数据；
4分：能正确输出第二组数据；
4分：能正确输出第三组数据；
4分：能正确输出第四组数据；
4分：能正确输出第五组数据；
4分：能正确输出第六组数据；
4分：能正确输出第七组数据；
4分：能正确输出第八组数据；
4分：能正确输出第九组数据；
4分：能正确输出第十组数据。
```

> **知识点考查：回文数判断；判断某个数是否包含x的方法**
>
> **思路解析：该题可以定义两个函数，一个函数用来判断当前遍历到的数是否为回文数，另一个函数用来判断当前得到的回文数是否包含7，当且仅当该数是回文数，并且此数当中包含7，这个数才需要统计**

参考代码

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

//cnt用来统计有多少个满足条件的数
int a, b, cnt;

//palindrome用来判断某个数是否为回文数
bool palindrome(int value) {
    //res用来存储value的逆序数
    int res = 0, temp = value;
    while(temp != 0) {
        res = res * 10 + temp % 10;
        temp /= 10;
    }
    return value == res;
}

//check用来检查判断某个数中是否包含7
bool check(int value) {
    while(value != 0) {
        if(value % 10 == 7)
            return 1;
```

```
        value /= 10;
    }
    return 0;
}

int main() {
    cin >> a >> b;
    //遍历a~b中的每一个数
    for(int i = a; i <= b; i++) {
        if(palindrome(i) && check(i))
            cnt++;
    }
    cout << cnt;
    return 0;
}
```

时间限制：1000MS，内存限制：65536KB

> **提示信息：**
>
> ABB形式的字符串：是由3个字符组成，其中后两个字符相同，第一个字符与后两个字符不同。
>
> 如:"cbb"、"q22"、"688"都是 ABB 形式的字符串；
>
> "abc"、"wwe"、"pop"都不是 ABB 形式的字符串。
>
> 子串：是指一个字符串中连续的一段字符序列。
>
> 如：字符串“Hello,World!"中，"Hello"、"ello"、"World"、"or"都是该字符串的子串。
>
> **（3）题目描述（编程实现）：**给定一个字符串S，请统计S中有多少个ABB形式的子串， 以及多少种ABB形式的子串。
>
> 例如：S=“nnnseebbetoosee”，ABB形式的子串有see、 ebb、too、see,共4个；不同子串有see、ebb、too,共3种。
>
> **输入描述：**输入一个长度不超过100的字符串S
>
> **输出描述：**输出两个整数，分别表示S中有多少个ABB形式的子串，以及多少种ABB形式的子串，整数之间以一个空格隔开
>
> **样例输入：**nnnseebbetoosee
>
> **样例输出：**4 3

```
5分：能正确输出第一组数据；
5分：能正确输出第二组数据；
5分：能正确输出第三组数据；
5分：能正确输出第四组数据；
5分：能正确输出第五组数据；
5分：能正确输出第六组数据；
5分：能正确输出第七组数据；
5分：能正确输出第八组数据；
5分：能正确输出第九组数据；
5分：能正确输出第十组数据。
```

> **知识点考查：字符串处理、set集合容器的应用**

> **思路解析：字符串长度不是很长，本题可以采用暴力枚举算法，截取要判断字符串的子串，该字串长度为3，然后判断当前这个字符是否为ABB形式，如当前字串x，那么当前仅当满足x[0]!=x[1] 且x[1]==x[2]的情况下，该子串是ABB的形式，若找到这样的子串，将其放入集合set中，且让统计ABB子串个数的计数变量+1，当遍历完给定的字符串之后，set集合中存储了字符串值不同的ABB形式的元素，直接输出集合大小**

参考代码

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

string value = "";
int cnt = 0;
//s集合不允许存储重复的字符串
set<string> s;
int main() {
    cin >> value;
    //如果字符串不足3个字符，那么肯定没有 ABB形式的字符串
    if (value.size() < 3) {
        cout << "0 0";
        return 0;
    }
    else {
        //遍历value[0]~value[n-3]的元素
        for(int i = 0;i < value.size() - 2;i++) {
            //从输入数据value中截取子串
            string v = value.substr(i,3);
            if(v[0] != v[1] && v[1] == v[2]) {
                s.insert(v);
                cnt++;
            }
        }
    }
    cout << cnt << " " << s.size();
    return 0;
}
```

时间限制：1000MS，内存限制：65536KB

> **（4）题目描述（编程实现）**：给定一个由n个整数组成的数列，请将其分割成左右两部分， 要求左半部分子数列的和与右半部分子数列的和最接近，请输出这两部分子数列和的差值（取非负值）。
>
> 例如：n=5，数列中的5个整数分别是2、1、3、4、3，将其分割成左右两部分，左半部分是2、1、3，右半部分是4、 3；此时两部分子数列的和最接近，差值为1。
>
> **输入描述**：
>
> 第一行输入一个整数n（2≤n≤100000）
>
> 第二行输入n个整数（1≤整数≤1000），整数之间以一个空格隔开
>
> **输出描述**：输出一个整数，表示这两部分子数列和的差值（取非负值）
>
> **样例输入**：

```
5
2 1 3 4 3
```

样例输出：

```
1
```

```
评分标准：
6分：能正确输出第一组数据；
6分：能正确输出第二组数据；
6分：能正确输出第三组数据；
6分：能正确输出第四组数据；
6分：能正确输出第五组数据；
6分：能正确输出第六组数据；
6分：能正确输出第七组数据；
6分：能正确输出第八组数据；
6分：能正确输出第九组数据；
6分：能正确输出第十组数据。
```

知识点考查：思维方法 、循环

思路解析：先计算出序列的总和，然后从第一个数开始遍历序列，计算第一个数，与后面n-1个数的差值，存储到变量minn中，minn用来存储差值最小数，然后继续遍历第二个数，将第二个数和第一个数累加，计算第一、二个数之和与后面n-2个数的差值，如果比minn中的差值小，那么存储到minn中，否则不做变动，依此类推，最终能够得到问题的解

参考代码

```cpp
#include<bits/stdc++.h>
using namespace std;
int n;
int a[100005];
int sum,tsum;
int minn = INT_MAX;
int main() {
    cin >> n;

    for(int i = 0; i < n; i++) {
        cin >> a[i];
        sum += a[i];
    }

    for(int i = 0; i < n; i++) {
        tsum += a[i];
        sum -= a[i];
        minn = min(abs(sum-tsum),minn);
    }

    cout << minn;

    return 0;
}
```

（5）题目描述（编程实现）：给定一个正整数n，请将n中的每位数字重新排列并组成一个新数，要求新数的值要小于n，请找出所有符合要求的新数中最大的那个正整数，如果不存在这样的正整数，则输出-1。

例1：n=312，312中每位上的数字依次是3、1、2，重新排列组成的新数有321、231、213、132、123，新数中小于312的有231、213、132、123，其中符合要求的最大正整数是231；

例2：n=123，123中每位上的数字依次是1、2、3，重新排列组成的新数有312、321、231、213、132，新数中不存在小于123的正整数，故输出-1。

输入描述：输入一个正整数 n (1≤ n <2的63次方)

输出描述：输出一个正整数，表示符合要求的最大正整数

样例输入：312

样例输出：231

```
8分：能正确输出第一组数据；
8分：能正确输出第二组数据；
8分：能正确输出第三组数据；
8分：能正确输出第四组数据；
8分：能正确输出第五组数据；
8分：能正确输出第六组数据；
8分：能正确输出第七组数据；
8分：能正确输出第八组数据；
8分：能正确输出第九组数据；
8分：能正确输出第十组数据。
```

知识点考查：全排列问题

思路解析：将n个字符按照任意方式排列，所得到的所有组合，叫做n的全排列。这些全排列从小到大的顺序，叫做这n个数的字典序。对于某一种排列方式，使用c++的stl库提供的prev_permutation可以获得其上一种字典序，也就是紧挨着他的，比他小的那种排列。因此只需要用字符串接受输入，然后获取该字符串的上一个字典序就行了。

参考代码

```
#include<bits/stdc++.h>
using namespace std;
int main() {
  string n;
  cin >> n;
  if (prev_permutation(n.begin(), n.end()))
    cout << n;
  else
    cout << -1;
}
```

（6）题目描述（编程实现）：靶场上有n块靶排成一排，从左到右依次编号为1、2、3、....n，且每块靶上都标有一个整数。当某块靶被击中后，击中者会得到 x * y * z 的积分。(y 表示被击中的靶上的数，x表示其左侧最近且未被击中的靶上的数，z表示其右侧最近且未被击中的靶上的数。如果其左侧不存在未被击中的靶，则x为1；如果其右侧不存在未被击中的靶，则z为1。）计算完积分后，这块靶就会退出靶场（不在这排靶中）。请计算击中所有靶后能得到的最高积分是多少?

例如：n=4，表示有4块靶，这4块靶上的数从左到右分别是3、2、4、6;

按照下列顺序打靶，可以得到最高积分：

1.打2号靶，得到的积分是24（`3*2*4`）；

2.打3号靶，得到的积分是72（`3*4*6`）；

3.打1号靶，得到的积分是18（`1*3*6`）；

4.打4号靶，得到的积分是6（`1*6*1`）；

最终获得的积分是120（24+72+18+6）。

输入描述：

第一行输入一个整数n（1≤n≤300），表示靶场上靶的数量

第二行输入n个整数（1≤整数≤100），分别表示从左到右每块靶上的数，整数之间以一个空格隔开

输出描述：输出一个整数，表示击中所有靶后能得到的最高积分

样例输入：

4

3 2 4 6

样例输出：120

```
评分标准：
10分：能正确输出第一组数据；
10分：能正确输出第二组数据；
10分：能正确输出第三组数据；
10分：能正确输出第四组数据；
10分：能正确输出第五组数据；
10分：能正确输出第六组数据；
10分：能正确输出第七组数据；
10分：能正确输出第八组数据；
10分：能正确输出第九组数据；
10分：能正确输出第十组数据。
```

知识点考查：搜索+回溯（暴力）、动态规划

思路解析1（搜索+回溯）：假设f（列表）=通过各种打靶顺序能够获得的的最大积分。那么从列表中去掉一个数nums[i]（打中一个靶子），按照题目规则，应当获得point=nums[i-1]×nums[i]×nums[i+1]的积分。显然，s+f(列表中剩下的数)的最大值=f(列表)。搜索每种结果即可，每次计算完一种情况下的最大值后，记得回溯，即将nums[i]插入到原来的位置。使用vector可以方便的进行删除与插入操作。这种方法的时间复杂度非常夸张，对测试用例可以得到正确结果，但是正式比赛提交必然超时

思路解析2（动态规划）：定义状态：dp[i][j] = x 表示：打掉 i 和 j 之间（不包括 i 和 j）的所有靶子，可以获得的最大积分为 x。状态转移方程：枚举i和j中间的每一个数k，i+1<=k<=j-1

```
dp[i][j] = max(dp[i][j], dp[i][k] + dp[k][j] + nums[k] * nums[i] * nums[j])
```

根据题目要求，我们可以在数组的最左、最右两端增加两个积分为1，且不能戳破的气球，并让i，j的范围在1~n之间变化，最后要求最大值就是dp[0][n+1]，也就是最左、最右两个我们增加的两个气球之间全打掉可以获得的最大值。

参考代码1（搜索+回溯）

```
#include <vector>
#include <algorithm>
#include <climits>
#include<iostream>
using namespace std;

int res = INT_MIN;

void backtrack(vector<int>& nums, int score) {
    if (nums.empty()) {
        res = max(res, score);
        return;
    }
    for (int i = 0; i < nums.size(); i++) {
        int point = (i - 1 < 0 ? 1 : nums[i - 1]) * nums[i] * (i + 1 >= nums.size() ? 1 : nums[i + 1]);//干掉第i个数的分数=左x自己x右
        vector<int> tempNums = nums;
        tempNums.erase(tempNums.begin() + i);//去掉第i个数
        backtrack(tempNums, score + point);//将分数累加，并搜索剩下的数能得到的最大值
        tempNums.insert(tempNums.begin() + i, nums[i]);//回溯
    }
}

int maxScore(vector<int>& nums) {
    res = INT_MIN;
    backtrack(nums, 0);
    return res;
}

int main() {
    int n, i;
    cin >> n;
    vector<int> nums(n);
    for (i = 0; i < n; i++)
    {
        cin >> nums[i];
    }
    int m = maxScore(nums);
    cout << m << endl;
    return 0;
}

```

参考代码2（动态规划）：

```
#include <vector>
#include <algorithm>
#include<iostream>
using namespace std;
int maxScore(vector<int>& nums) {
  int n = nums.size();
  vector<int> points(n + 2);
```

```
  points[0] = 1;
  points[n + 1] = 1;
  for (int i = 0; i < n; ++i) {
    points[i + 1] = nums[i];
  }

  vector<vector<int>> dp(n + 2, vector<int>(n + 2, 0));

  for (int length = 1; length <= n; ++length) {
    for (int i = 0; i <= n - length; ++i) {
      int j = i + length + 1;
      for (int k = i + 1; k < j; ++k) {
        dp[i][j] = max(dp[i][j], dp[i][k] + dp[k][j] + points[i] * points[k] * points[j]);
      }
    }
  }
  return dp[0][n + 1];
}
int main()
{
  int n;
  vector<int> v(n);
  cin >> n;
  int i;
  for (i = 0; i < n; i++)
  {
    cin >> v[i];
  }
  cout<<maxScore(v);
}

```

三、特别鸣谢

> 在此特别感谢**Silicon（邹老师）**提供最后两题压轴题的思路、标程，如果有疑问或者需要，可以联系他。