常用模型练习

人员

左子毅、朱奕鸣、杨洋、刘子淇、赵清航、周子航、于迦浩、刘佳赫 到课

作业检查

左子毅 已完成

朱奕鸣 已完成

杨洋 已完成

刘子淇 已完成

赵清航 已完成

周子航 已完成

于迦浩 已完成

刘佳赫 已完成

作业

https://www.luogu.com.cn/contest/172789

A、B、C、D、E 题必做

要求 A、B、C 题都可以5分钟之内写出来，下周要求默写

课堂表现

同学们课上表现都比较好，今天课上讲的知识比较多，同学们课下要好好复习

课堂内容

P1020 [NOIP1999 提高组] 导弹拦截

vector中lower_bound和upper_bound的用法:

```
vector<int>::iterator it = lower_bound(vec.begin(), vec.end(), x);  // it是vector中第一个大于等于x的数的迭代器
vector<int>::iterator it = upper_bound(vec.begin(), vec.end(), x);  // it是vector中第一个大于x的数的迭代器
int pos = lower_bound(vec.begin(), vec.end(), x) - vec.begin();  // pos是vector中第一个大于等于x的位置的下标
int pos = upper_bound(vec.begin(), vec.end(), x) - vec.begin();  // pos是vector中第一个大于x的位置的下标
```

数组中的用法:

```
int pos = lower_bound(a+1, a+n+1, x) - a;  // pos是数组中第一个大于等于x的位置的下标
int pos = upper_bound(a+1, a+n+1, x) - a;  // pos是数组中第一个大于x的位置的下标
```

lower_bound和upper_bound默认在升序序列中进行查找
可以通过自定义cmp函数，实现lower_bound和upper_bound在降序序列中的二分查找

```
bool cmp(int a, int b) {
  return a > b;
}
int pos = lower_bound(a+1, a+n+1, x, cmp) - a;  // pos是数组中第一个小于等于x的位置的下标
int pos = upper_bound(a+1, a+n+1, x, cmp) - a;  // pos是数组中第一个小于x的位置的下标
```

```
// 重载运算符
struct node {
  int a, b;
  friend bool operator < (node p, node q) {
    return p.a < q.a;
  }
  bool operator < (const node& p) const {
    return a < p.a;
  }
};
```

```cpp
#include <bits/stdc++.h>

using namespace std;

bool cmp(int a, int b) { return a > b; }

int main() {
  vector<int> vv;
  int x;
  while (cin >> x) vv.push_back(x);

  vector<int> vec1, vec2;
  for (int i : vv) {
    if (vec1.empty() || i<=vec1.back()) vec1.push_back(i);
    else *upper_bound(vec1.begin(), vec1.end(), i, cmp) = i;

    if (vec2.empty() || i>vec2.back()) vec2.push_back(i);
    else *lower_bound(vec2.begin(), vec2.end(), i) = i;
  }

  cout << vec1.size() << endl;
  cout << vec2.size() << endl;
  return 0;
}
```

P1106 删数问题

类似于单调栈问题，利用vector维护一个单调上升的栈
当要插入一个新元素时，如果新元素比栈顶的值小，并且可以删除的话，那么删除栈顶的值

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
  string s; cin >> s;
  int k; cin >> k;

  vector<char> vec;
  for (char i : s) {
    while (k && (!vec.empty() && i<vec.back())) {
      vec.pop_back();
      --k;
    }
    vec.push_back(i);
  }

  while (k) { vec.pop_back(); --k; }

  bool flag = false;
  for (char i : vec) {
    if (i != '0') flag = true;
    if (flag) cout << i;
  }
  if (!flag) cout << 0;
  cout << endl;
  return 0;
}
```

CF1883D In Love

```cpp
multiset<int> s;
multiset 是不去重的set，会自动从小到大排序
从小到大输出:
for (auto it = s.begin(); it != s.end(); it++) cout << *it << endl;
从大到小输出:
for (auto it = s.rbegin(); it != s.rend(); it++) cout << *it << endl;

set中的最小值: *s.begin()
set中的最大值: *s.rbegin()

s.erase(5): 删除set中所有的5
s.erase(s.find(5)): 只删除一个5
```

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
  int T; cin >> T;
  multiset<int> sL, sR;
  while (T -- ) {
    char op[2];
    int l, r;
    cin >> op >> l >> r;
    if (op[0] == '+') {
      sL.insert(l); sR.insert(r);
    } else {
      sL.erase(sL.find(l)); sR.erase(sR.find(r));
    }

    if (!sL.empty() && *sL.rbegin()>*sR.begin()) cout << "YES" << endl;
    else cout << "NO" << endl;
  }
  return 0;
}
```

T454708 计算 A*B + C

```cpp
#include <bits/stdc++.h>

using namespace std;

typedef __int128 LL;

LL read() {
    char ch = getchar();
    LL res = 0, f = 1;
    while (!isdigit(ch)) {
        if (ch == '-') f = -1;
        ch = getchar();
    }
    while (isdigit(ch)) res = res*10 + (ch-'0'), ch = getchar();
    return res*f;
}

void print(LL x) {
    if (x < 0) { putchar('-'); x = -x; }
    if (x > 9) print(x/10);
    putchar(x%10 + '0');
}

int main()
{
    LL a, b, c;
    a = read(), b = read(), c = read();
    print(a*b + c);
  return 0;
}
```