上 海 交 通 大 学 试 卷

(2019 至 2020 学年第一学期)

班级号: _____ 学号: _____ 姓名: _____

课程名称: 程序设计 成绩: _____

注意事项

1. 本场考试考试形式为上机考试，考试时间 180 分钟，满分 300 分。
2. 考试过程中，请遵守上海交通大学学生考试纪律规定，并禁止使用任何存储设备，违者将被视为作弊。
3. 如有任何疑问，请举手示意。
4. 请注意提交代码的代码风格！极为糟糕的代码风格（例如改变语法的某些宏定义）会酌情扣分！
5. 最终答案提交方法为提交源代码, 各题目源代码文件名见下, **所有题目只需要提交 hpp 头文件**。所有源代码放在桌面上**名为学号**的文件夹内，**不建子目录**。
6. 每道题同时会提供一个 cpp 文件，你可以使用编译该文件来测试运行自己的代码，题目描述中的输入格式和输出格式均指该文件，你也可以更改这个文件以方便调试自己的代码，但造成的后果由自己承担。

   例如：对于第一题，在 code1.cpp 文件内已经 `#include "code1.hpp"`，你可以编译运行 code1.cpp，并按照样例输入来测试自己完成的 code1.hpp。

   特别需要注意的是此文件以及输入、输出等均只与调试有关，与最终测试无关。
7. 请严格按照题目描述的算法解决问题，否则将不能得到分数。
8. 存在内存泄漏的点将扣除 20% 的分数，请注意检查空间的释放。
9. 在所有题目中均不可以在 hpp 文件中添加新的头文件

题目基本信息

题目名称	绝世好题	圆	OJ 的继承与派生
分值	100	100	100
源代码文件名	code1.hpp	code2.hpp	code3.hpp
时间限制	1s	1s	1s
内存限制	512M	512M	512M

1 绝世好题

1.1 题目描述

想必各位身经百战的同学都已经熟练掌握了如何写一个链表，数组的使用更是不在话下。下面我们就来写一个以链表为元素的数组吧！

你需要完成一个链表数组类型 `LinkListArray`，里面包含 `len` 个保存正整数的单链表（你可以自由决定是否带头结点），并要求在任意时刻每个链表里的元素都是满足非严格升序排列的（可以有存在相同的元素），也就是说，在任意时刻，每个单链表中的第 0 个元素应该是这个单链表中非严格最小的元素，第 1 个元素应该是这个单链表中非严格次小的元素……链表中允许元素重复。开始时，每一个单链表中都没有元素。你应该支持以下操作（所有数均为 0-base）：

- 0 在第 s_1 个链表中加入数字 s_2
- 1 查询第 s_1 个链表中第 s_2 小的元素，如果第 s_1 个链表中的元素数小于 s_2，则返回 -1。
- 2 将第 s_1 个链表与第 s_2 个链表合并成一个升序链表，保存为第 s_1 个链表，将第 s_2 个链表清空 ($s_1 \neq s_2$)。

对于如何合并两个有序链表得到一个新的有序链表，你应该使用如下算法：

1. 如果两个链表都为空，则算法结束。
2. 如果一个链表非空，另外一个链表为空，则将非空链表的最小元素从链表中取出，加入新链表的末尾。回到第 1 步。
3. 如果两个链表皆非空，比较两个链表各自的最小元素，将两个元素较小的那个取出，加入新链表的末尾。回到第 1 步。

这样就得到了新的有序链表。当然，你应该自己填补算法的细节。然而善良的助教已经写好了完整的 `LinkListArray` 类，你只需要填补它所依赖的 `LinkList` 类就好啦！

注意事项

- 你应该把链表中的每一个节点封装为一个类/结构体。每个节点包含一个数据元素和一个后继指针。
- 你应该把每一个单链表也封装成为一个结构体。
- 为防止内存泄漏，你需要在程序最后释放所有申请的堆空间。
- 请填补给的样例代码来通过本题。

1.2 输入格式

第一行两个数 len 和 m 表示链表数组长度和总操作数。

接下来的 m 行，每行包含第一个数 op 表示操作类型，后面紧跟两个正整数，表示题目中的 s_1, s_2。保证操作合法且所给部分和所给要求不会产生超时问题。

1.3 输出格式

对于每个操作 1，输出一行一个正整数，表示查询的结果。

1.4 样例输入

```
4 9
0 0 5
0 0 3
1 0 0
1 0 2
0 1 4
2 1 0
1 1 1
1 1 0
1 0 0
```

1.5 样例输出

```
3
-1
4
3
-1
```

1.6 数据范围与约束

对于 30% 的数据，$len = 1$。

对于 60% 的数据，不包含操作 2。

对于 100% 的数据，$1 \leqslant len \leqslant 100$，数组中元素小于等于 10^9。

2 圆

2.1 题目描述

结构体 `Circle` 表示一个圆，(x, y) 是圆心的坐标，`r` 是圆的半径。`CircleSet` 类表示一个圆的集合，类中有一个数组 `circles[]` 来存储集合中所有圆，**集合中所有圆的半径互不相同。以下描述中，圆 p 指的是 `circles[p]`。(0-base)**

你需要实现以下函数：

- `CircleSet` 类中的析构函数。
- `CircleSet` 类中的下标运算符重载 operator[]：设 `set` 是 CircleSet 类的对象，`set[p]` 返回圆 p 的半径。
- `CircleSet` 类中的函数 `bool checkContaining(int p, int q)`：如果圆 p 严格包含(不相切) 圆 q，返回 1，否则返回 0。
- `CircleSet` 类中的函数 `int getCircleContainingQ(int q)`：返回集合中严格包含圆 q 的半径最小的圆的半径。(保证一定存在)

注意事项

- 圆 p 严格包含圆 q 的充要条件：圆 p 的半径大于圆 q 的半径且两圆心的距离小于两圆半径之差的绝对值。
- 如果你需要比较 a 和 $\sqrt{b}$ 的大小，为了避免浮点误差，建议你比较 a^2 和 b 的大小。
- 不能在类中定义新的成员变量，不能修改类中已经给定的函数。

2.2 输入格式

第一行有两个整数 n 和 m。

之后有 m 行，每行的第一个整数 type 表示测试类型。

type = 1 时，测试析构函数。

type = 2 时，测试下标运算符重载。

type = 3 时，type 后面有两个整数 p 和 q，测试函数 bool checkContaining(int p, int q)。

type = 4 时，type 后面有一个整数 q，测试函数 int getCircleContainingQ(int q)。

保证一个测试文件中的 type 都相等。

2.3 输出格式

具体输出格式请看主函数。

2.4 样例输入

```
10 10
0 0 1
0 0 9
0 0 2
0 0 6
0 0 8
0 0 7
9 0 12
−9 0 13
0 9 10
0 −9 11
1
2
3 0 1
3 1 0
3 5 6
3 9 0
3 9 2
4 0
4 2
4 5
```

2.5 样例输出

```
1 9 2 6 8 7 12 13 10 11
0
1
0
1
0
```

```
2
6
8
```

2.6 数据范围和约定

为了分别评测每个函数的实现，每个测试点只对应一个函数。一共有 20 个测试点来测试你的程序。

圆的横坐标、纵坐标和半径都是 [-10000,10000] 范围内的整数。

测试点编号	type	n	m	Hint	分值
1 – 4	1	100	1	测试析构函数	20%
5 – 8	2	100	1	测试下标运算符重载	20%
9 – 14	3	100	10	测试函数 `bool checkContaining(int p, int q)`	30%
15 – 20	4	100	10	测试函数 `int getCircleContainingQ(int q)`	30%

3 OJ 的继承与派生

3.1 题目背景

经过一个学期小作业和机考的洗礼，相信大家对 onlinejudge 有了一定的了解。终于到了期末机考，这次是没机会拿 onlinejudge 当调试器了（误），但摆在你面前的是一个实现简单 onlinejudge 派生类的题目。

给你的是一个 `BaseJudger` 基类，现在需要你完成的是 `ICPCJudger` 派生类、`OIJudger` 派生类和 `SpacialJudger` 派生类。实现的目的很简单，就是能够用它们对提交的 `time`，`memory`，`output` 在不同 judge 机制下进行评分。

3.2 题目描述

给了你一个基类 `BaseJudger`，其中你只需要在`// todo` 的地方补全构造函数和析构函数即可，其他地方不允许进行任何修改，不允许添加任何成员和成员函数。（下发代码中的 `size_t` 是一种数据类型，你可以把它当作非负的 `int`）

题目的重点是需要你在此基础上**派生**出 `class ICPCJudger`，`class OIJudger` 和 `class SpacialJudger`。

派生类中请根据需要添加你需要的成员以及成员函数。

ICPCJudger

`ICPCJudger` 的构造方式，`judger = new ICPCJudger(time_limit, memory_limit, answer);`

`ICPCJudger` 的 `Submit` 评测标准是：

当 `time <= time_limit && memory <= memory_limit` 并且答案正确时给 100 分，其余都给零分；

可以多次 `Submit` 并且取最高的分数；

答案正确的要求是 `output` 与 `answer` 完全相同；

OIJudger

`OIJudger` 的构造方式，`judger = new OIJudger(time_limit, memory_limit, answer);`

`OIJudger` 的 `Submit` 评测标准是：

当 time <= time_limit && memory <= memory_limit 并且答案正确时给 100 分，其余都给零分；

只能 Submit 一次取唯一分数，一旦多次提交直接爆零。

注意：不同于 ICPCJudger 的是，OIJudger 对答案正确的要求是忽略 output 和 answer 行尾的空格的（如果 answer 是多行的，则忽略每一行行尾空格）；

SpacialJudger

SpacialJudger 的构造方式,judger = new SpacialJudger(time_limit, memory_limit, full_score_time, full_score_memory, answer);

SpacialJudger 的 Submit 评测标准：

当用 SpacialJudger 评测的时候,会额外给出两个指标 full_score_time 和 full_score_memory；

除了要求答案正确以外，对于 time 指标，当 time <= full_score_time 时 time_score 给 100 分，当 time >= time_limit 时 time_score 指标给 0 分，对于 full_score_time < time < time_limit 的情况，time_score 与 time 是线性关系，对于 memory 指标是同理的（time_score 和 memory_score 的计算均向下取整),并且最后的分数是由 floor(time_score * memory_score / 100) 得到的。（也就是说，如果 time 和 memory 都是 50 分，最后的得分是 25 分）

可以多次 Submit 并且取最高的分数。

对答案正确的要求同 ICPCJudger 一样；

3.3 输入格式

输入的第一行是 judger_name

输入的第二行给出了在当前 judger 下需要给出的若干项指标

输入的接下去若干行是正确的 answer（保证多行 answer 之间没有空行）

输入一行 ##### 分割符，以判断 answer 读取完毕

接下去的输入都是提交，可能会重复多次。

每次提交的第一行是 time 和 memory

每次提交的接下去若干行是 output(保证多行 output 之间没有空行）

每次提交的最后一行是 ##### 分割符，以判断一次提交结束

3.4 输出格式

一行一个数，即最终的 `score`

3.5 样例输入

```
ICPCJudger
1000 1000
abcd
dcba
#####
500 1001
abcd
dcba
#####
500 500
abcd
dcba
#####
```

3.6 样例输出

```
100
```

3.7 数据范围与约定

1-4 测试点，`judger_name = ICPCJudger`

5-7 测试点，`judger_name = SpacialJudger`

8-10 测试点，`judger_name = OIJudger`

测试数据中保证分数始终为整数，且能被整除。