Во всех предлагаемых задачах вам необходимо написать ТОЛЬКО класс или классы, предусмотренные условием. Файл, отправляемый вами на проверку в систему Ejudge, НЕ ДОЛЖЕН (!) содержать никаких директив `#include`, `using`, не должен использовать никакие возможности библиотек, не должен содержать функцию `main`.

Для того, чтобы проверить ваши классы на локальной машине, опишите главную функцию **в отдельном файле**, а тот файл, который вы намерены отправить на проверку, подключите директивой `#include`.

Все описываемые вами классы, методы и переменные должны иметь в точности те имена, которые указаны в задании; в противном случае проверяющая система их отвергнет.

Задача может быть засчитана только в том случае, если она проходит все предусмотренные для неё в системе тесты; при этом текст вашего решения проверяют преподаватели. Прохождение тестов является для засчитывавния задачи условием необходимым, но не достаточным. Достаточным условием является полное соответствие решения заданию.

1 *(5 баллов).* Опишите класс `MyInt`, хранящий в закрытом поле целое число, получающий это число через параметр конструктора и возвращающий его из метода `Get`, который должен допускать работу для константных объектов.

2 *(5 баллов).* Модифицируйте класс `MyInt` из предыдущей задачи так, чтобы при создании копии объекта этого класса (например, при передаче его по значению в функции и т. п.) число, хранящееся в объекте-копии, было на единицу больше числа, хранившегося в объекте-оригинале.

3 *(10 баллов).* Снабдите класс `MyInt` из двух предыдущих задач операциями «`+`» и «`+=`», имеющими естественный смысл.

4 *(10 баллов).* Опишите класс `Counted`, имеющий метод `GetCount`, который допускает вызов без объекта, по имени класса (например, должен быть допустимым оператор «`x = Counted::GetCount();`»). Класс должен допускать создание объекта без указания параметров. Метод `GetCount` должен в любой момент времени возвращать целое число, соответствующее количеству существующих в настоящий момент объектов класса `Counted`. **Не забудьте про случай создания объекта как копии другого объекта!**

5 *(25 баллов).* Опишите класс `SpecialInts`, представляющий понятие «упорядоченного набора целых чисел», реализованный через массив с изменяемой при необходимости длиной (вводить явные ограничения на длину массива запрещается).

Определите для этого класса операцию «запятая» (`operator,()`), которая будет добавлять в набор ещё одно число (если таковое удовлетворяет условию, заданному функцией `Check`; в противном случае число не добавляется) и возвращать ссылку на объект, для которого она вызвана, чтобы можно было применить несколько таких операций подряд. Создайте операцию индексирования

(`operator[]()`) для извлечения заданного элемента набора и метод `Len()`, выдающий текущую длину набора.

При обращении к элементам набора через операцию индексирования проверяйте корректность заданного индекса; если он меньше нуля или, наоборот, слишком большой (в наборе нет элемента с этим номером) — выбрасывайте исключение класса `BadRange`; метод `Get()` класса `BadRange` должен возвращать значение индекса, вызвавшее ошибку.

Вся выделенная динамическая память должна быть корректно освобождена.

6 *(20 баллов).* Модифицируйте класс `SpecialInts` из предыдущей задачи, снабдив его виртуальной функцией `Check`, принимающей параметр типа `int` и возвращающей логическое значение. В классе `SpecialInts` функция `Check` должна всегда возвращать «истину», но предполагается, что классы-наследники могут заменить эту функцию своей версией таковой. При добавлении нового числа в набор это число должно проверяться вызовом функции `Check`, и если функция вернула истину, число должно помещаться в набор, в противном случае — игнорироваться.

Унаследуйте от класса `SpecialInts` класс `SmallInts`, принимающий для хранения только числа, не превосходящие по модулю 10. Этот класс должен отличаться от базового только функцией `Check`. Учтите, что в ходе тестирования от класса SpecialInts будут наследоваться также другие потомки.

7 *(25 баллов).* Опишите абстрактный класс `AbstractMax`, представляющий понятие «максимум среди целых чисел» в соответствии с отношением порядка (полного или частичного), которое задаётся чисто виртуальным методом

```
virtual bool Order(int a, int b) const
```

(метод возвращает истину, если `a` предшествует `b` в смысле водимого отношения порядка). Класс должен помнить текущее значение максимума (начальное значение максимума задаётся параметром конструктора) и реализовывать (невиртуальный) метод `Consider`, через который объекту класса передаётся очередное целое число. Необходимо предусмотреть также метод `Get` для извлечения значения текущего максимума из объекта.

Унаследуйте от класса `AbstractMax` классы `NativeMax` (обычное отношение порядка для целых, то есть `Order` возвращает `true`, если первый аргумент строго меньше второго), и `DeciMax` (числа упорядочиваются по остатку от деления на `10`: `Order` возвращает `true`, если остаток от деления на `10` первого аргумента строго меньше остатка от деления на `10` второго аргумента). Учтите, что в процессе тестирования от вашего класса будут наследоваться и другие потомки.