Сборщик мусора

В качестве задания предлагается реализовать сборку мусора в упрощенном варианте. Сборщик мусора должен автоматически освобождать память, которая недоступна для пользователя посредством указателей (т.е. разруливать ситуации, в которых без сборщика мусора была бы утечка памяти).

Сборщик мусора будет работать из следующих предположений:

1. Объекты, выделенные на стеке, удаляются автоматически.
2. Любой объект, контролируемый сборщиком мусора, должен быть во владении некоторого другого объекта, находящегося под его контролем.

Таком образом, все объекты можно разделить на два типа: выделенные на стеке и выделенные динамически и принадлежащие какому-либо другому объекту. Следовательно, можно сформулировать критерий достижимости объекта:

1. Если объект выделен на стеке, он достижим.
2. Если до объекта можно добраться путем перехода по владеемым объектам, начиная с некоторого достижимого объекта, то этот объект достижим.

Пометить все достижимые объекты можно запустив любой обход графа (например, поиск в глубину) из объектов, выделенных на стеке. После того, как все достижимые объекты будут помечены, все остальные можно удалить, тем самым освободив память.

Задание:

1. Реализовать абстрактный класс **SmartObject**, с интерфейсом:

```
#pragma once

#include <vector>

class SmartObject
{
public:
    SmartObject();

    virtual std::vector<SmartObject *> pointers() const = 0; //
Should be implemented by user
    void* operator new(size_t size);

    virtual ~SmartObject();
};
```

SmartObject должен обладать следующим функционалом: при создании объекта объекта он должен быть зарегистрирован в сборщике мусора, с пометкой, на стеке он или в куче. При вызове деструктора он, соответственно, должен быть снят с учета в сборщике мусора. Добиться такого поведения можно путем переопределения операторов **new** (и, возможно, **delete**) и конструктора и деструктора.

2. Реализовать класс **GarbageCollector**, который будет являться синглтоном. Класс должен предоставлять возможность регистрировать, “дерегистрировать” объекты и запускать сборщик мусора.
3. При попытке динамически создать объект должно проверяться, что количество выделенной памяти не больше, чем некоторая максимально доступная память. В противном случае запускается сборка мусора, и повторно проверяется, можем ли мы создать объект. Если нет, то кидается исключение std::bad_alloc.
4. Написать тесты, в которых бы явным образом создавались недостижимые объекты, в частности, ссылающиеся друг на друга. При запуске этих тестов под **valgrind** утечек памяти быть не должно.