Задължително домашно #2

Напишете програма с проста имплементация на гарбидж колектор, който управлява заделянето и освобождаването на памет.
Изисквания:

- Програмата трябва да може да се компилира под Линукс с g++ -Wall
- Разделянето на кода в отделния файлове е задължително
- Компилация с make е препоръчителна

Неща, които носят бонуси:

- Компилация без грешки и уорнинги
- Компилация с make
- Добре подреден, форматиран и именуван код

1. Дефинирайте статичен клас `GC`, който:
    a. има скрит дефолтен конструктор за да не могат да се правят инстанции
    b. всички атрибути, които добавите са скрити
    c. има статичен метод `void *allocate(size_t size)`, който получава размер в байтове и връща указател към заделен блок от памет
    d. има статичен метод `void free(void *region)`, който получава указател към блок от памет и го освобождава
    e. има статичен метод `void grow(void *region, size_t size)`, който получава указател към блок от памет и нов размер и увеличава блока
    f. има статичен метод `void shrink(void *region, size_t size)`, който получава указател към блок от памет и нов размер и намалява блока
    g. има статичен метод `void dump(string file_path)`
2. Дефинирайте базов клас `BaseObject`, който:
    a. има предефиниран оператор new, който използва метода allocate на `GC`
    b. има предефиниран оператор delete, който използва метода free на `GC`
    c. аналогични предефинирани оператори `new[]` и `delete[]`
3. Демонстрирайте работата на `GC` като изпълните следните демонстрации. За всяка демонстрация заделете памет, задайте стойност, изпринтирайте стойността, освободете паметта и направете dump след заделяните и след освобождаването.
    a. заделяне за 1 примитивна променлива

```
int *a = GC::allocate(sizeof(int))
*a = 5;
cout << a << " " << *a << endl;
GC::free(a);
a = NULL;
```

    b. заделяне за масив от примитивни примитивен тип

```
int *a = GC::allocate(sizeof(int) * 5)
```

c. заделяне за клас, който наследява `BaseObject`, което ще му позволи да използва неговите предефинирани оператори. Наследяването може да стане със синтаксиса `class Child : public Parent { ... }`

```
Test *test = new Test()
delete test;
```

d. заделяне за масив от клас, който наследява `BaseObject`

4. Работа на гарбидж колектора
    a. в класа трябва да пазите информация за заделените блокове памет. Можете да изберете под каква форма и с каква структура, но задължително трябва да можете за всеки заделен блок да изброите указателя към първия му байт и размера на блока
    b. когато програмата се стартира се заделя памет от **1024** байта. В `GC` се пази указател към нея(налична памет) и това е паметта, в която може да заделя блокове
    c. когато се извика allocate се намира и връща адреса на първия байт от наличната памет, който е свободен и е последван от **size - 1** свободни байтове(непрекъсната област от **size** свободни байта). Записва се информацията за областта и тя вече се третира като заделена - `allocate` и `grow` не може да заделят нова памет в нея докато не бъде освободена
    d. когато се извика `free` и се подаде валиден указател към първия байт на заделен блок този блок се освобождава - информацията за него се изтрива/изчиства/нулира (в зависимост как менажирате тази информация)
    e. когато се извика `grow` и са подадени валиден указател към първия байт на заделен блок и нов размер, който е по-голям от размера на блока и след този блок има достатъчно свободни байтове то информацията за блока се обновява да отрази новия му размер
    f. когато се извика `shrink` и са подадени валиден указател към първия байт на заделен блок и нов размер, който е по-малък от размера на блока то информацията за блока се обновява да отрази новия му размер
    g. да се хвърлят подходящи изключения когато:
        i. `allocate` не може да намери достатъчно голям свободен блок
        ii. на `free`, `grow` и `shrink` не е подаден указател към първи байт от заделен блок
        iii. на `grow` се подаде размер, по-малък от текущия
        iv. на `shrink` се подаде размер, по-голям от текущия
        v. `grow` не може да увеличи размера на блока
5. Метод `dump` - трябва да запише информацията за състоянието на паметта и заделените блокове във файл на подадения път. Форматът е:

```
<брой заделени байтове> / <размер на цялата налична памет>
<адрес на първи байт от наличната памет> - <адрес на последния байт от наличната памет>
```

```
<начален адрес на блок 1> - <краен адрес на блок 1> (<размер в байтове>)
...
<начален адрес на блок N> - <краен адрес на блок N> (<размер в байтове>)
```

Пример:

```
14 / 1024
0x123456 - 0x123459 (4)
0x12345A - 0x123461 (8)
0x123452 - 0x123463 (2)
```

6. За 6
    a. паметта да бъде организирана под формата на страници от по 1024 байта. Страниците не са последователни в паметта и може да се намират на произволни адреси
    b. когато се извика `allocate` и няма достатъчно голяма свободна област да се задели нова страница и да се върне блок в нея
    c. максималният брой страници да бъде **5**