```
/* The following code example is taken from the book
 * "The C++ Standard Library - A Tutorial and Reference, 2nd Edition"
 * by Nicolai M. Josuttis, Addison-Wesley, 2012
 *
 * (C) Copyright Nicolai M. Josuttis 2012.
 * Permission to copy, use, modify, sell and distribute this software
 * is granted provided this copyright notice appears in all copies.
 * This software is provided "as is" without express or implied
 * warranty, and with no claim as to its suitability for any purpose.
 */
#include <memory>     // for shared_ptr
#include <sys/mman.h> // for shared memory
#include <fcntl.h>
#include <unistd.h>
#include <cstring>    // for strerror()
#include <cerrno>     // for errno
#include <string>
#include <iostream>

class SharedMemoryDetacher
{
  public:
    void operator () (int* p) {
        std::cout << "unlink /tmp1234" << std::endl;
        if (shm_unlink("/tmp1234") != 0) {
            std::cerr << "OOPS: shm_unlink() failed" << std::endl;
        }
    }
};

std::shared_ptr<int> getSharedIntMemory (int num)
{
    void* mem;
    int shmfd = shm_open("/tmp1234", O_CREAT|O_RDWR, S_IRWXU|S_IRWXG);
    if (shmfd < 0) {
        throw std::string(strerror(errno));
    }
    if (ftruncate(shmfd, num*sizeof(int)) == -1) {
        throw std::string(strerror(errno));
    }
    mem = mmap(nullptr, num*sizeof(int), PROT_READ | PROT_WRITE,
               MAP_SHARED, shmfd, 0);
    if (mem == MAP_FAILED) {
        throw std::string(strerror(errno));
    }
    return std::shared_ptr<int>(static_cast<int*>(mem),
                                SharedMemoryDetacher());
}

int main()
{
    // get and attach shared memory for 100 ints:
    std::shared_ptr<int> smp(getSharedIntMemory(100));

    // init the shared memory
    for (int i=0; i<100; ++i) {
```

```
        smp.get()[i] = i*42;
    }

    // deal with shared memory somewhere else:
    //...
    std::cout << "<return>" << std::endl;
    std::cin.get();

    // release shared memory here:
    smp.reset();
    //...
}
```