```cpp

#include <exception>
#include <vector>
#include <iostream>
#include <algorithm>
#include <memory>
using std::cout;
using std::endl;
using std::vector;
using std::shared_ptr;

class BadInput {
};

const int MIN  = 0;

template <class T>
vector<T> slice(std::vector<T> vec, int start, int step, int stop);

//q5 part 5.1
template <class T>
vector<T> slice(std::vector<T> vec, int start, int step, int stop)
{
    if(start < MIN || start >= vec.size()){
        throw BadInput();
    }
    if(stop < MIN || stop > vec.size()){
        throw BadInput();
    }
    if(step <= MIN){
        throw BadInput();
    }
    if(start >= stop){
        vector<T> empty_vector;
        return empty_vector;
    }
    vector<T> new_vector;
    for(int it = start; it < stop; it+=step){
        T to_add = vec[it];
        new_vector.push_back(to_add);
    }
    return new_vector;
}

```

```cpp
//q5 part 5.2
class A {
public:
    vector<shared_ptr<int>> values;
    void add(int x){
        shared_ptr<int> to_add(new int(x));
        values.push_back(to_add);
    }
};

int main() {
    A a, sliced;
    a.add(0); a.add(1); a.add(2); a.add(3); a.add(4); a.add(5);
    sliced.values = slice(a.values, 1, 1, 4);
    *(sliced.values[0]) = 800;
    std::cout << *(a.values[1]) << std::endl;
    return 0;
}

```