```cpp
#include <vector>
#include <exception>
#include <iostream>
#include <memory>

class BadInput : public std::exception{};

//Check if input is invalid
bool isNotInputValid(int vec_size, int start, int step, int stop){
    return start < 0 || stop < 0 || step <= 0 || start >= vec_size || stop > vec_size;
}

template <class T>
std::vector<T> slice(std::vector<T> vec, int start, int step, int stop){
    if(isNotInputValid(vec.size(), start, step, stop)){
        throw BadInput();
    }
    if(start >= stop){
        return std::vector<T>();
    }
    std::vector<T> new_vec = std::vector<T>();
    for(int i = start; i < stop; i+=step){
        new_vec.push_back(vec[i]);
    }
    return new_vec;
}

// By using shared_ptr all allocated memory is safely freed.
// For this reason we coulden't use int*.
// Also, we've added constructors as instructed.
class A {
    public:
    A() = default;
    A(const A&) = default;
    ~A() = default;
    std::vector<std::shared_ptr<int>> values;
    void add(int x) { values.push_back(std::shared_ptr<int>(new int(x))); }
};

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;
}
```