

```

1
2  #include <exception>
3  #include <vector>
4  #include <iostream>
5  #include <algorithm>
6  #include <memory>
7  using std::cout;
8  using std::endl;
9  using std::vector;
10 using std::shared_ptr;
11
12 class BadInput {
13 };
14
15 const int MIN = 0;
16
17 template <class T>
18 vector<T> slice(std::vector<T> vec, int start, int step, int stop);
19
20 //q5 part 5.1
21 template <class T>
22 vector<T> slice(std::vector<T> vec, int start, int step, int stop)
23 {
24     if(start < MIN || start >= vec.size()){
25         throw BadInput();
26     }
27     if(stop < MIN || stop > vec.size()){
28         throw BadInput();
29     }
30     if(step <= MIN){
31         throw BadInput();
32     }
33     if(start >= stop){
34         vector<T> empty_vector;
35         return empty_vector;
36     }
37     vector<T> new_vector;
38     for(int it = start; it < stop; it+=step){
39         T to_add = vec[it];
40         new_vector.push_back(to_add);
41     }
42     return new_vector;
43 }
44

```

```
45 //q5 part 5.2
46 class A {
47 public:
48     vector<shared_ptr<int>> values;
49     void add(int x){
50         shared_ptr<int> to_add(new int(x));
51         values.push_back(to_add);
52     }
53 };
54
55 int main() {
56     A a, sliced;
57     a.add(0); a.add(1); a.add(2); a.add(3); a.add(4); a.add(5);
58     sliced.values = slice(a.values, 1, 1, 4);
59     *(sliced.values[0]) = 800;
60     std::cout << *(a.values[1]) << std::endl;
61     return 0;
62 }
63
64
```