#1

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
  vector<int> numbers;

  numbers.push_back(10);
  numbers.push_back(20);
  numbers.push_back(30);

  cout << "First element: " << numbers[0] << endl;
  cout << "Second element using at(): " << numbers.at(1) << endl;

  numbers.pop_back();

  cout << "Size after pop_back: " << numbers.size() << endl;

  for (int i = 0; i < numbers.size(); i++) {
    cout << "Element at index " << i << ": " << numbers[i] << endl;
  }
  return 0;
}
```

```
First element: 10
Second element using at(): 20
Size after pop_back: 2
Element at index 0: 10
Element at index 1: 20
```

#2

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
  vector<int> vec = {1,2,3,4,5};

  for (auto it = vec.begin(); it != vec.end(); it++) {
    cout << *it << endl;
  }
  return 0;
}
```

```
1
2
3
4
5
```

#3

```
#include <iostream>
#include <memory>
using namespace std;
int main() {
  unique_ptr<int> p1 = make_unique<int>(10);
  cout << "Value inside unique_ptr: " << *p1 << endl;
  return 0;
}
```

```
Value inside unique_ptr: 10
```

#4

```
#include <iostream>
#include <memory>
using namespace std;
int main() {
  shared_ptr<int> p1 = make_shared<int>(10);
  shared_ptr<int> p2 = p1;
  cout << "Value inside shared_ptr p1: " << *p1 << endl;
  cout << "Value inside shared_ptr p2: " << *p2 << endl;
  return 0;
}
```

```
Value inside shared_ptr p1: 10
Value inside shared_ptr p2: 10
```

#5

```
#include <iostream>
#include <memory>
using namespace std;
int main() {
  unique_ptr<int> p1 = make_unique<int>(10);
  cout << "Value (the memory address) of p1 = " << p1 << endl;
  cout << "Value inside unique_ptr: " << *p1 << endl;
```

```
  return 0;
}
```

```
./main.cpp:6:49: error: invalid operands to binary expression ('basic_ostream<char, char_traits<char>>
' and 'unique_ptr<int>')
    6 |   cout << "Value (the memory address) of p1 = " << p1 << endl;
      |   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ^  ~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/cstddef:124:5: note: candida
te function template not viable: no known conversion from 'basic_ostream<char, char_traits<char>>' to
'byte' for 1st argument
  124 |     operator<<(byte __b, _IntegerType __shift) noexcept
      |     ^          ~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/system_error:339:5: note: ca
ndidate function template not viable: no known conversion from 'unique_ptr<int>' to 'const error_code'
 for 2nd argument
  339 |     operator<<(basic_ostream<_CharT, _Traits>& __os, const error_code& __e)
      |     ^                                               ~~~~~~~~~~~~~~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/ostream:564:5: note: candida
te function template not viable: no known conversion from 'unique_ptr<int>' to 'char' for 2nd argument
  564 |     operator<<(basic_ostream<_CharT, _Traits>& __out, char __c)
      |     ^                                                ~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/ostream:570:5: note: candida
te function template not viable: no known conversion from 'unique_ptr<int>' to 'char' for 2nd argument
  570 |     operator<<(basic_ostream<char, _Traits>& __out, char __c)
      |     ^                                              ~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/ostream:581:5: note: candida
te function template not viable: no known conversion from 'unique_ptr<int>' to 'signed char' for 2nd a
rgument
  581 |     operator<<(basic_ostream<char, _Traits>& __out, signed char __c)
      |     ^                                              ~~~~~~~~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/ostream:586:5: note: candida
te function template not viable: no known conversion from 'unique_ptr<int>' to 'unsigned char' for 2nd
 argument
  586 |     operator<<(basic_ostream<char, _Traits>& __out, unsigned char __c)
      |     ^                                              ~~~~~~~~~~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/ostream:662:5: note: candida
te function template not viable: no known conversion from 'unique_ptr<int>' to 'const char *' for 2nd
argument
  662 |     operator<<(basic_ostream<char, _Traits>& __out, const char* __s)
      |     ^                                              ~~~~~~~~~~~~~~~
/nix/store/14c6s4xzhy14l2b05s00rjns2j93gzz4-gcc-13.2.0/include/c++/13.2.0/ostream:675:5: note: candida
te function template not viable: no known conversion from 'unique_ptr<int>' to 'const signed char *' f
or 2nd argument
  675 |     operator<<(basic_ostream<char, _Traits>& __out, const signed char* __s)
      |     ^                                              ~~~~~~~~~~~~~~~~~~~~~~
```

#6

```
#include <iostream>
#include <memory>
using namespace std;
int main() {
  unique_ptr<int> p1 = make_unique<int>(10);

  cout << "Value (the memory address) of p1 = " << p1.get() << endl;

  cout << "Value inside unique_ptr: " << *p1 << endl;
  return 0;
}
```

```
Value (the memory address) of p1 = 0x557e35f472b0
Value inside unique_ptr: 10
```

#7

```
#include <iostream>
using namespace std;
int main() {
  auto sum = [](int a, int b) {
```

```
    return a + b;
  };
  cout << "Sum of 3 and 4: " << sum(3, 4) << endl;
  return 0;
}
```

```
Sum of 3 and 4: 7
```

#8

```
#include <iostream>
using namespace std;
int main() {
  int x = 10;
  auto printX = [x]() {
    cout << "Captured x: " << x << endl;
  };
  printX();
  return 0;
}
```

```
Captured x: 10
```

#9

```
#include <iostream>
#include <stdexcept>
using namespace std;
int main() {
  int x = -1;
  if (x < 0) {
    throw invalid_argument("Negative number error");
  }
  return 0;
}
```

```
terminate called after throwing an instance of 'std::invalid_argument'
  what():  Negative number error
```

#10

```
#include <iostream>
```

```
#include <stdexcept>
using namespace std;
int main() {
  try {
   throw runtime_error("An error occurred");
  } catch (const exception& e) {
    cout << e.what() << endl;
  }
  return 0;
}
```

```
An error occurred
```