Project 1: It's Time for Dodger Baseball! Testing

There were 100 test cases. Each test was worth 1.3 points each; to run the test cases:

1. Remove the main routine from your `BaseballRoster.cpp` file.
2. Append the following text to the end of your `BaseballRoster.cpp` file and build the resulting program.
3. For any test case you wish to try, run the program, providing as input the test number.

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <cassert>
#include <vector>
#include <type_traits>
#include "BaseballRoster.h"

using namespace std;

bool findPlayer2type(bool (BaseballRoster::*)(const PType&,
NType&) const) { return true; }
bool findPlayer2type(bool (BaseballRoster::*)(const PType&,
NType&)) { return false; }
bool findPlayer2type(...) { return false; }
bool findPlayer3type(bool (BaseballRoster::*)(int, PType&,
NType&) const) { return true; }
bool findPlayer3type(bool (BaseballRoster::*)(int, PType&,
NType&)) { return false; }
bool findPlayer3type(...) { return false; }

PType SOMENAME = PType("Y Z");
PType DEFAULTNAME = PType();
PType ARRAYNAME[6] = {
      std::string("A B"), std::string("C D"), std::string("E
F"),
      std::string("G H"), std::string("I J"), std::string("K L")
};

NType SOMEVALUE = -1;
NType DEFAULTV = NType();
NType ARRAYV[6] = {
      13, 23, 33, 43, 53, 63
};

bool has(const BaseballRoster& m, PType& name, const NType& v)
{
```

```
	NType v2 = DEFAULTV;
	m.findPlayer(name, v2);
	NType v3 = SOMEVALUE;
	m.findPlayer(name, v3);
	return v2 == v && v3 == v;
}

void testone(int n)
{
	BaseballRoster m;
	switch (n)
	{
	default: {
		cout << "Bad argument" << endl;
	} break; case  1: {

	assert((is_same<decltype(&BaseballRoster::noPlayers), bool
(BaseballRoster::*)() const>::value));
	} break; case  2: {

	assert((is_same<decltype(&BaseballRoster::numberOfPlayers)
, int (BaseballRoster::*)() const>::value));
	} break; case  3: {

	assert((is_same<decltype(&BaseballRoster::playerOnRoster),
bool (BaseballRoster::*)(const PType&) const>::value));
	} break; case  4: {

	assert(findPlayer2type(&BaseballRoster::findPlayer));
	} break; case  5: {

	assert(findPlayer3type(&BaseballRoster::findPlayer));
	} break; case  6: {
		assert(m.noPlayers());
	} break; case  7: {
		assert(m.numberOfPlayers() == 0);
	} break; case  8: {
		assert(!m.updatePlayer(DEFAULTNAME, SOMEVALUE) &&
m.numberOfPlayers() == 0);
	} break; case  9: {
		assert(!m.dfa(DEFAULTNAME) && m.numberOfPlayers() ==
0);
	} break; case 10: {
		assert(!m.playerOnRoster(DEFAULTNAME));
	} break; case 11: {
		NType v = SOMEVALUE;
```

```
            assert(!m.findPlayer(DEFAULTNAME, v) && v ==
SOMEVALUE);
      } break; case 12: {
            NType v = SOMEVALUE;
            assert(!m.findPlayer(0, DEFAULTNAME, v) && v ==
SOMEVALUE);
      } break; case 13: {
            assert(m.addPlayer(SOMENAME, SOMEVALUE));
      } break; case 14: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            assert(!m.noPlayers());
      } break; case 15: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            assert(m.numberOfPlayers() == 1);
      } break; case 16: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            assert(m.playerOnRoster(SOMENAME));
      } break; case 17: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            NType v = DEFAULTV;
            assert(m.findPlayer(SOMENAME, v));
      } break; case 18: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            NType v = DEFAULTV;
            m.findPlayer(SOMENAME, v);
            assert(v == SOMEVALUE);
      } break; case 19: {
            m.addPlayer(ARRAYNAME[0], SOMEVALUE);
            NType v = DEFAULTV;
            assert(!m.findPlayer(ARRAYNAME[1], v));
      } break; case 20: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            NType v = SOMEVALUE;
            m.findPlayer(ARRAYNAME[1], v);
            assert(v == SOMEVALUE);
      } break; case 21: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            PType n = DEFAULTNAME;
            NType v = DEFAULTV;
            assert(m.findPlayer(0, n, v));
      } break; case 22: {
            m.addPlayer(SOMENAME, SOMEVALUE);
            PType n = DEFAULTNAME;
            NType v = DEFAULTV;
            m.findPlayer(0, n, v);
            assert(n == SOMENAME && v == SOMEVALUE);
      } break; case 23: {
```

```
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(!m.noPlayers() && m.numberOfPlayers() == 2);
      } break; case 24: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(m.playerOnRoster(ARRAYNAME[0]) &&
m.playerOnRoster(ARRAYNAME[1]));
      } break; case 25: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(has(m, ARRAYNAME[0], ARRAYV[0]) && has(m,
ARRAYNAME[1], ARRAYV[1]));
      } break; case 26: {
            m.addPlayer(ARRAYNAME[0], SOMEVALUE);
            m.addPlayer(ARRAYNAME[1], SOMEVALUE);
            assert(has(m, ARRAYNAME[0], SOMEVALUE) && has(m,
ARRAYNAME[1], SOMEVALUE));
      } break; case 27: {
            assert(m.addPlayer(ARRAYNAME[0], ARRAYV[0]));
            assert(m.addPlayer(ARRAYNAME[1], ARRAYV[1]));
      } break; case 28: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[0], ARRAYV[2]);
            assert(m.numberOfPlayers() == 2);
      } break; case 29: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[0], ARRAYV[2]);
            assert(has(m, ARRAYNAME[0], ARRAYV[0]) && has(m,
ARRAYNAME[1], ARRAYV[1]));
      } break; case 30: {
            assert(m.addPlayer(ARRAYNAME[0], ARRAYV[0]));
            assert(m.addPlayer(ARRAYNAME[1], ARRAYV[1]));
            assert(!m.addPlayer(ARRAYNAME[0], ARRAYV[2]));
      } break; case 31: {
            assert(m.addPlayer(ARRAYNAME[0], ARRAYV[0]));
            assert(m.addPlayer(ARRAYNAME[1], ARRAYV[1]));
            assert(!m.addPlayer(ARRAYNAME[0], ARRAYV[0]));
      } break; case 32: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.updatePlayer(ARRAYNAME[1], SOMEVALUE);
            assert(m.numberOfPlayers() == 3 &&
m.playerOnRoster(ARRAYNAME[0]) &&
```

```
                    m.playerOnRoster(ARRAYNAME[1]) &&
m.playerOnRoster(ARRAYNAME[2]));
      } break; case 33: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.updatePlayer(ARRAYNAME[1], SOMEVALUE);
            assert(has(m, ARRAYNAME[0], ARRAYV[0]) && has(m,
ARRAYNAME[1], SOMEVALUE) &&
                    has(m, ARRAYNAME[2], ARRAYV[2]));
      } break; case 34: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            assert(m.updatePlayer(ARRAYNAME[1], SOMEVALUE));
      } break; case 35: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.updatePlayer(ARRAYNAME[2], ARRAYV[0]);
            assert(m.numberOfPlayers() == 2 && has(m,
ARRAYNAME[0], ARRAYV[0]) &&
                    has(m, ARRAYNAME[1], ARRAYV[1]) &&
!m.playerOnRoster(ARRAYNAME[2]));
      } break; case 36: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(!m.updatePlayer(ARRAYNAME[2], ARRAYV[2]) &&
!m.updatePlayer(ARRAYNAME[3], ARRAYV[0]));
      } break; case 37: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addOrUpdate(ARRAYNAME[1], ARRAYV[1]);
            assert(!m.noPlayers() && m.numberOfPlayers() == 2);
      } break; case 38: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addOrUpdate(ARRAYNAME[1], ARRAYV[1]);
            assert(has(m, ARRAYNAME[0], ARRAYV[0]) && has(m,
ARRAYNAME[1], ARRAYV[1]));
      } break; case 39: {
            m.addPlayer(ARRAYNAME[0], SOMEVALUE);
            m.addOrUpdate(ARRAYNAME[1], SOMEVALUE);
            assert(has(m, ARRAYNAME[0], SOMEVALUE) && has(m,
ARRAYNAME[1], SOMEVALUE));
      } break; case 40: {
            assert(m.addPlayer(ARRAYNAME[0], ARRAYV[0]));
            assert(m.addOrUpdate(ARRAYNAME[1], ARRAYV[1]));
      } break; case 41: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
```

```
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addOrUpdate(ARRAYNAME[0], ARRAYV[2]);
            assert(m.numberOfPlayers() == 2 && has(m,
ARRAYNAME[0], ARRAYV[2]) &&
                  has(m, ARRAYNAME[1], ARRAYV[1]));
      } break; case 42: {
            assert(m.addPlayer(ARRAYNAME[0], ARRAYV[0]));
            assert(m.addPlayer(ARRAYNAME[1], ARRAYV[1]));
            assert(m.addOrUpdate(ARRAYNAME[0], ARRAYV[2]));
      } break; case 43: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.addOrUpdate(ARRAYNAME[1], SOMEVALUE);
            assert(m.numberOfPlayers() == 3 && has(m,
ARRAYNAME[0], ARRAYV[0]) &&
                  has(m, ARRAYNAME[1], SOMEVALUE) && has(m,
ARRAYNAME[2], ARRAYV[2]));
      } break; case 44: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            assert(m.addOrUpdate(ARRAYNAME[1], SOMEVALUE));
      } break; case 45: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addOrUpdate(ARRAYNAME[2], ARRAYV[0]);
            assert(m.numberOfPlayers() == 3 && has(m,
ARRAYNAME[0], ARRAYV[0]) &&
                  has(m, ARRAYNAME[1], ARRAYV[1]) && has(m,
ARRAYNAME[2], ARRAYV[0]));
      } break; case 46: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(m.addOrUpdate(ARRAYNAME[2], ARRAYV[2]));
      } break; case 47: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(m.dfa(ARRAYNAME[1]));
      } break; case 48: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.dfa(ARRAYNAME[1]);
            assert(!m.noPlayers() && m.numberOfPlayers() == 1 &&
has(m, ARRAYNAME[0], ARRAYV[0]) &&
                  !m.playerOnRoster(ARRAYNAME[1]));
      } break; case 49: {
```

```
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.dfa(ARRAYNAME[0]);
            assert(!m.noPlayers() && m.numberOfPlayers() == 1 &&
has(m, ARRAYNAME[1], ARRAYV[1]) &&
                  !m.playerOnRoster(ARRAYNAME[0]));
      } break; case 50: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.dfa(ARRAYNAME[0]);
            m.dfa(ARRAYNAME[1]);
            assert(m.numberOfPlayers() == 0);
      } break; case 51: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.dfa(ARRAYNAME[1]);
            m.dfa(ARRAYNAME[2]);
            m.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            assert(m.numberOfPlayers() == 2 && has(m,
ARRAYNAME[0], ARRAYV[0]) &&
                  has(m, ARRAYNAME[3], ARRAYV[3]) &&
!m.playerOnRoster(ARRAYNAME[1]) &&
                  !m.playerOnRoster(ARRAYNAME[2]));
      } break; case 52: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            assert(!m.dfa(ARRAYNAME[2]) && m.numberOfPlayers()
== 2);
      } break; case 53: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            PType n;
            NType v;
            assert(!m.findPlayer(-1, n, v));
      } break; case 54: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            PType n = SOMENAME;
            NType v = SOMEVALUE;
            m.findPlayer(-1, n, v);
            assert(n == SOMENAME && v == SOMEVALUE);
      } break; case 55: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            PType n;
            NType v;
```

```
            assert(!m.findPlayer(2, n, v));
      } break; case 56: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            PType n = SOMENAME;
            NType v = SOMEVALUE;
            m.findPlayer(2, n, v);
            assert(n == SOMENAME && v == SOMEVALUE);
      } break; case 57: {
            m.addPlayer(DEFAULTNAME, SOMEVALUE);
            assert(m.numberOfPlayers() == 1 && has(m,
DEFAULTNAME, SOMEVALUE));
      } break; case 58: {
            m.updatePlayer(DEFAULTNAME, SOMEVALUE);
            assert(m.numberOfPlayers() == 0 &&
!m.playerOnRoster(DEFAULTNAME));
      } break; case 59: {
            m.addOrUpdate(DEFAULTNAME, SOMEVALUE);
            assert(m.numberOfPlayers() == 1 && has(m,
DEFAULTNAME, SOMEVALUE));
      } break; case 60: {
            m.addPlayer(DEFAULTNAME, SOMEVALUE);
            m.dfa(DEFAULTNAME);
            assert(m.numberOfPlayers() == 0 &&
!m.playerOnRoster(DEFAULTNAME));
      } break; case 61: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m.swapRoster(m2);
                  assert(m.numberOfPlayers() == 3);
            }
      } break; case 62: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m.swapRoster(m2);
                  assert(has(m, ARRAYNAME[1], ARRAYV[1]) &&
has(m, ARRAYNAME[2], ARRAYV[2]) &&
```

```
                        has(m, ARRAYNAME[3], ARRAYV[3]) &&
!m.playerOnRoster(ARRAYNAME[0]));
            }
      } break; case 63: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m.swapRoster(m2);
                  assert(m2.numberOfPlayers() == 2);
            }
      } break; case 64: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m.swapRoster(m2);
                  assert(has(m2, ARRAYNAME[0], ARRAYV[0]) &&
has(m2, ARRAYNAME[1], ARRAYV[1]) &&
                        !m2.playerOnRoster(ARRAYNAME[2]) &&
!m2.playerOnRoster(ARRAYNAME[3]));
            }
      } break; case 65: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
                  m.swapRoster(m2);
                  assert(m.numberOfPlayers() == 3 &&
m2.numberOfPlayers() == 5);
            }
      } break; case 66: {
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
```

```
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  assert(m2.numberOfPlayers() == 2 &&
m2.playerOnRoster(ARRAYNAME[1]) &&
!m2.playerOnRoster(ARRAYNAME[3]));
            }
      } break; case 67: {
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2.dfa(ARRAYNAME[1]);
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.dfa(ARRAYNAME[2]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m2.dfa(ARRAYNAME[0]);
                  m2.dfa(ARRAYNAME[3]);
                  m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            }
            assert(true);  // no corruption so bad that
destruction failed
      } break; case 68: {
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  BaseballRoster m3(m2);
                  m3.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m3.dfa(ARRAYNAME[1]);
                  m3.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m3.dfa(ARRAYNAME[2]);
                  m3.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m3.dfa(ARRAYNAME[0]);
                  m3.dfa(ARRAYNAME[3]);
                  m3.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            }
            assert(true);  // no corruption so bad that
destruction failed
      } break; case 69: {
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  BaseballRoster m3(m2);
                  assert(m3.numberOfPlayers() == 3);
            }
```

```
	} break; case 70: {
		m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
		m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
		m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
		{
			BaseballRoster m2(m);
			m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
			assert(m2.numberOfPlayers() ==
m.numberOfPlayers() + 1);
		}
	} break; case 71: {
		m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
		m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
		m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
		{
			BaseballRoster m2(m);
			m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
			assert(m2.numberOfPlayers() == 4 &&
m2.playerOnRoster(ARRAYNAME[1]) &&
m2.playerOnRoster(ARRAYNAME[3]));
		}
	} break; case 72: {
		m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
		m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
		m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
		{
			BaseballRoster m2(m);
			m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
			assert(m2.numberOfPlayers() == 4 &&
m2.playerOnRoster(ARRAYNAME[1]) &&
!m2.playerOnRoster(ARRAYNAME[4]));
		}
	} break; case 73: {
		{
			BaseballRoster m2;
			m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
			m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
			m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
			BaseballRoster m3;
			m3.addPlayer(ARRAYNAME[4], ARRAYV[4]);
			m3.addPlayer(ARRAYNAME[5], ARRAYV[5]);
			m3 = m2;
			assert(m3.numberOfPlayers() == 3 &&
m2.numberOfPlayers() == 3);
		}
	} break; case 74: {
		m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
```

```
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
                  m2 = m;
                  m2.addPlayer(ARRAYNAME[5], ARRAYV[5]);
                  assert(m2.numberOfPlayers() ==
m.numberOfPlayers() + 1);
            }
      } break; case 75: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
                  m2.addPlayer(ARRAYNAME[5], ARRAYV[5]);
                  m2 = m;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  assert(m2.playerOnRoster(ARRAYNAME[0]) &&
                        m2.playerOnRoster(ARRAYNAME[1]) &&
                        m2.playerOnRoster(ARRAYNAME[2]) &&
                        !m2.playerOnRoster(ARRAYNAME[3]));
            }
      } break; case 76: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
                  m2.addPlayer(ARRAYNAME[5], ARRAYV[5]);
                  m2 = m;
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  assert(m.playerOnRoster(ARRAYNAME[0]) &&
                        m.playerOnRoster(ARRAYNAME[1]) &&
                        !m.playerOnRoster(ARRAYNAME[2]));
            }
      } break; case 77: {
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2 = m2;
```

```
                  assert(m2.numberOfPlayers() == 3);
                  assert(m2.playerOnRoster(ARRAYNAME[0]) &&
                        m2.playerOnRoster(ARRAYNAME[1]) &&
                        m2.playerOnRoster(ARRAYNAME[2]));
            }
            assert(true);  // no corruption so bad that
destruction failed
      } break; case 78: {
            {
                  BaseballRoster m2;
                  m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
                  m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
                  m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
                  m2 = m2;
                  m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
                  assert(m2.playerOnRoster(ARRAYNAME[0]) &&
                        m2.playerOnRoster(ARRAYNAME[1]) &&
                        m2.playerOnRoster(ARRAYNAME[2]) &&
                        m2.playerOnRoster(ARRAYNAME[3]));
            }
      } break; case 79: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            BaseballRoster m3;
            combineRosters(m, m2, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]));
      } break; case 80: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            BaseballRoster m3;
            combineRosters(m2, m, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]));
      } break; case 81: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
```

```
            BaseballRoster m3;
            combineRosters(m, m2, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]) &&
                  m3.playerOnRoster(ARRAYNAME[3]) &&
                  m3.playerOnRoster(ARRAYNAME[4]));
      } break; case 82: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            BaseballRoster m3;
            assert(combineRosters(m, m2, m3));
      } break; case 83: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            BaseballRoster m3;
            m3.addPlayer(ARRAYNAME[2], ARRAYV[5]);
            combineRosters(m, m2, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]) &&
                  m3.playerOnRoster(ARRAYNAME[3]) &&
                  m3.playerOnRoster(ARRAYNAME[4]) &&
                  has(m3, ARRAYNAME[2], ARRAYV[2]) &&
                  !has(m3, ARRAYNAME[2], ARRAYV[5]));
      } break; case 84: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            BaseballRoster m3;
            m3.addPlayer(ARRAYNAME[2], ARRAYV[5]);
            assert(combineRosters(m, m2, m3));
      } break; case 85: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
```

```
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m3;
            combineRosters(m, m2, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]) &&
                  m3.playerOnRoster(ARRAYNAME[3]));
      } break; case 86: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m3;
            assert(combineRosters(m, m2, m3));
      } break; case 87: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m2.addPlayer(ARRAYNAME[3], ARRAYV[4]);
            BaseballRoster m3;
            combineRosters(m, m2, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]));
      } break; case 88: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m2.addPlayer(ARRAYNAME[3], ARRAYV[4]);
            BaseballRoster m3;
            assert(!combineRosters(m, m2, m3));
      } break; case 89: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            combineRosters(m, m2, m);
```

```
            assert(m.playerOnRoster(ARRAYNAME[0]) &&
                  m.playerOnRoster(ARRAYNAME[1]) &&
                  m.playerOnRoster(ARRAYNAME[2]) &&
                  m.playerOnRoster(ARRAYNAME[3]) &&
                  m.playerOnRoster(ARRAYNAME[4]));
      } break; case 90: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[3], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[4], ARRAYV[4]);
            combineRosters(m, m2, m2);
            assert(m2.playerOnRoster(ARRAYNAME[0]) &&
                  m2.playerOnRoster(ARRAYNAME[1]) &&
                  m2.playerOnRoster(ARRAYNAME[2]) &&
                  m2.playerOnRoster(ARRAYNAME[3]) &&
                  m2.playerOnRoster(ARRAYNAME[4]));
      } break; case 91: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            BaseballRoster m3;
            outright(m, m2, m3);
            assert(!m3.playerOnRoster(ARRAYNAME[0]) &&
                  m3.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]));
      } break; case 92: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            BaseballRoster m3;
            outright(m, m2, m3);
            assert(m3.playerOnRoster(ARRAYNAME[0]) &&
                  !m3.playerOnRoster(ARRAYNAME[1]) &&
                  m2.playerOnRoster(ARRAYNAME[1]) &&
                  m3.playerOnRoster(ARRAYNAME[2]));
      } break; case 93: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
```

```
            outright(m, m2, m);
            assert(m.playerOnRoster(ARRAYNAME[0]) &&
                  !m.playerOnRoster(ARRAYNAME[1]) &&
                  m.playerOnRoster(ARRAYNAME[2]));
      } break; case 94: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.addPlayer(ARRAYNAME[4], ARRAYV[3]);
            m.addPlayer(ARRAYNAME[3], ARRAYV[2]);
            BaseballRoster m2;
            outright(m, m2, m2);
            assert(m2.playerOnRoster(ARRAYNAME[0]) &&
                  m2.playerOnRoster(ARRAYNAME[1]) &&
                  m2.playerOnRoster(ARRAYNAME[2]) &&
                  m2.playerOnRoster(ARRAYNAME[3]) &&
                  m2.playerOnRoster(ARRAYNAME[4]));
      } break; case 95: {
            m.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m.addPlayer(ARRAYNAME[2], ARRAYV[3]);
            m.addPlayer(ARRAYNAME[3], ARRAYV[2]);
            BaseballRoster m2;
            outright(m2, m, m);
            assert(m2.noPlayers());
      } break; case 96: {
            BaseballRoster m2;
            m2.addPlayer(ARRAYNAME[0], ARRAYV[0]);
            m2.addPlayer(ARRAYNAME[1], ARRAYV[1]);
            m2.addPlayer(ARRAYNAME[2], ARRAYV[2]);
            m2.addPlayer(ARRAYNAME[2], ARRAYV[3]);
            m2.addPlayer(ARRAYNAME[3], ARRAYV[2]);
            outright(m, m2, m2);
            assert(m2.noPlayers());
      } break; case 97: {
            BaseballRoster m2 = m;
            BaseballRoster m3;
            outright(m, m2, m3);
            assert(m3.numberOfPlayers() == m.numberOfPlayers());
      } break; case 98: {
            BaseballRoster m2;
            BaseballRoster m3(m);
            outright(m2, m3, m3);
            assert(m3.noPlayers());
      } break; case 99: {
            BaseballRoster m2;
```

```
            outright(m2, m2, m2);
            assert(m2.numberOfPlayers() == 0);
      } break; case 100: {
            const int NITEMS = 2000;
            for (int k = 0; k < NITEMS; k++)
                  assert(m.addPlayer(std::to_string(k),
SOMEVALUE));
            assert(m.numberOfPlayers() == NITEMS);
      }
      }
}

int main()
{
    cout << "Enter test number: ";
    int n;
    cin >> n;
    testone(n);
    cout << "Passed" << endl;
}
```