```cpp
#include "regular.h"

int Regular::m_cp_times = 0, Regular::m_def_times = 0;

Regular::Regular()
{
  cout << "    Default Constructor Called (" << ++m_def_times
       << " times)" << endl;
  m_example_member = new int;
  *m_example_member = 0;
}

Regular::Regular(const int example_member): Regular()
{
  cout << "    Parameterized Constructor Called" << endl;
  *m_example_member = example_member;
}

Regular::Regular(const Regular& ex): Regular()
{
  cout << "    Copy Constructor (" << ++m_cp_times
       << " times)" << endl;
  *m_example_member = *ex.m_example_member;
}

Regular::~Regular()
{
  delete m_example_member;
  m_example_member = nullptr;
}

Regular& Regular::operator=(const Regular& ex)
{
  cout << "    Copy Assignment Operator" << endl;
  *m_example_member = *ex.m_example_member;
  return *this;
}

void Regular::get_summary()
{
  cout << "    Total Copies: " << m_cp_times
       << "    Total Default Calls: " << m_def_times << endl;
}
```

regular.cpp

```cpp
#include "movesemantics.h"

int MoveSemantics::m_cp_times = 0, MoveSemantics::m_def_times = 0;

MoveSemantics::MoveSemantics()
{
  cout << "    Default Constructor Called (" << ++m_def_times
       << " times)" << endl;
  m_example_member = new int;
  *m_example_member = 0;
}

MoveSemantics::MoveSemantics(const int example_member):
     MoveSemantics()
{
  cout << "    Parameterized Constructor Called" << endl;
  *m_example_member = example_member;
}

MoveSemantics::MoveSemantics(MoveSemantics&& ex)
{
  cout << "    Move Constructor" << endl;
  m_example_member = ex.m_example_member;
  ex.m_example_member = nullptr;
  // nullptr is the new C++11 constant for null pointers.
}

MoveSemantics::MoveSemantics(const MoveSemantics& ex):
     MoveSemantics()
{
  cout << "    Copy Constructor (" << ++m_cp_times
       << " times)" << endl;
  *m_example_member = *ex.m_example_member;
}

MoveSemantics::~MoveSemantics()
{
  delete m_example_member;
  m_example_member = nullptr;
}

MoveSemantics& MoveSemantics::operator=(MoveSemantics&& ex)
{
  cout << "    Move Assignment Operator" << endl;
  m_example_member = ex.m_example_member;
  ex.m_example_member = nullptr;
  return *this;
}

MoveSemantics& MoveSemantics::operator=(const MoveSemantics& ex)
{
  cout << "  Copy Assignment Operator" << endl;
  *m_example_member = *ex.m_example_member;
  return *this;
}

void MoveSemantics::get_summary()
{
  cout << "    Total Copies: " << m_cp_times
       << "    Total Default Calls: " << m_def_times << endl;
}
```

movesemantics.cpp

```cpp
#include <iostream>
#include "movesemantics.h"
#include "regular.h"
using namespace std;

template <class T>
T some_func(T in) { return in; }

template <class T>
void run_example()
{
  cout << "  First Part" << endl;
  T b;
  T c(some_func<T>(b));

  cout << endl << "  Second Part" << endl;
  b = some_func<T>(c);

  cout << endl << "  Third Part" << endl;
  b = move(c);
  T d(move(b));

  cout << endl << "  Fourth Part" << endl;

  int i = 5;
  T e(i);
  e = 4;

  cout << "  TOTALS: " << endl;
  T::get_summary();
}

int main()
{
  cout << "No Move Semantics Example: " << endl << endl;
  run_example<Regular>();
  cout << endl << endl << "With Move Semantics: " << endl << endl;
  run_example<MoveSemantics>();
  return 0;
}
```

example.cpp

```
$ /usr/bin/g++ -g -Wall -W -pedantic-errors -std=c++11 *.cpp
$ ./example

No Move Semantics Example:

  First Part
    Default Constructor Called (1 times)
    Default Constructor Called (2 times)
    Copy Constructor (1 times)
    Default Constructor Called (3 times)
    Copy Constructor (2 times)

  Second Part
    Default Constructor Called (4 times)
    Copy Constructor (3 times)
    Default Constructor Called (5 times)
    Copy Constructor (4 times)
    Copy Assignment Operator

  Third Part
    Copy Assignment Operator
    Default Constructor Called (6 times)
    Copy Constructor (5 times)

  Fourth Part
    Default Constructor Called (7 times)
    Parameterized Constructor Called
    Default Constructor Called (8 times)
    Parameterized Constructor Called
    Copy Assignment Operator
  TOTALS:
    Total Copies: 5    Total Default Calls: 8


With Move Semantics:

  First Part
    Default Constructor Called (1 times)
    Default Constructor Called (2 times)
    Copy Constructor (1 times)
    Move Constructor

  Second Part
    Default Constructor Called (3 times)
    Copy Constructor (2 times)
    Move Constructor
    Move Assignment Operator

  Third Part
    Move Assignment Operator
    Move Constructor

  Fourth Part
    Default Constructor Called (4 times)
    Parameterized Constructor Called
    Default Constructor Called (5 times)
    Parameterized Constructor Called
    Move Assignment Operator
  TOTALS:
    Total Copies: 2    Total Default Calls: 5
```

output.txt