**Logitech**

**HW06**

1. Pass the tests as follow, please make sure there is no memory leak:

TEST1

{

Coffee \*mycoffee = new BlackCoffee ;

EXPECT\_EQ (mycoffee.get\_cost(), 10);

EXPECT\_STREQ(mycoffee.get\_content(), “black coffee”);

Mycoffee = new MilkDecorator (mycoffee);

EXPECT\_EQ (mycoffee.get\_cost(), 15);

EXPECT\_STREQ(mycoffee.get\_content(), “black coffee with milk”);

Mycoffee = new SugarDecorator (mycoffee);

EXPECT\_EQ (mycoffee.get\_cost(), 16);

EXPECT\_STREQ(mycoffee.get\_content(), “black coffee with milk with sugar”);

}

TEST2

{

Coffee \*mycoffee = new BlackCoffee ;

Mycoffee = new SugarDecorator (mycoffee);

EXPECT\_EQ (mycoffee.get\_cost(), 11);

EXPECT\_STREQ(mycoffee.get\_content(), “black coffee with sugar”);

}

1. Please write a linked list data structure with Boost iterator (don’t use std::list, or you would have to use boost::iterator\_adapter for instead). Design your own unit test of several use cases before you implement the details.
2. Pass the following tests:

TEST1

{

int variableA (10);

int variableB (5);

std::vector<Command\*> commands;

command.emplace\_back (new SimpleCommand(&variableA, std::bind (&divide, variableB)) );

command.emplace\_back (new SimpleCommand(&variableA, std::bind (&add, variableB)) );

EXPECT\_EQ (variableA, 10);

for (auto& c : command)

c.execute();

EXPECT\_EQ (variableA, 7);

}

TEST2

{

int variableA (10);

int variableB (5);

MacroCommand commands;

commands.push\_back (new SimpleCommand(&variableA, std::bind (&divide, variableB)) );

commands.push\_back (new SimpleCommand(&variableA, std::bind (&add, variableB)) );

EXPECT\_EQ (variableA, 10);

commands.execute();

EXPECT\_EQ (variableA, 7);

}