CS 126: C++ Object Review

Fill in as many blanks as you can. Feel free to work with your neighbors.

ExpressionValue.h

```
#_____EXPRESSIONVALUE_H
#define EXPRESSIONVALUE_H

_________ cs126 {
   class ExpressionValue {
   _______:
      double value_;  // not a blank

   public:
      ExpressionValue();
      ExpressionValue(double value);

      double getValue() ________;
      void setValue(double value);
   };
}

#endif //EXPRESSIONVALUE_H
```

ExpressionValue.cpp

```
#include _____ExpressionValue.h_____

double cs126::ExpressionValue::getValue() const {
   return value_;
}

void cs126::ExpressionValue::setValue(double value) {
   ___________________________________
}

cs126::ExpressionValue::ExpressionValue(double value) : _______(value) {}

cs126::ExpressionValue::ExpressionValue() : ExpressionValue(0.0) {}
```

main.cpp

```
#include ___iostream___
#include ___ExpressionValue.h___

______ std::cout;
______ std::endl;
______ cs126::ExpressionValue;

int _________() {
   cout << "Hello, World!" << endl;

   ExpressionValue eValue(7.0);
   eValue___setValue(14.0);
   cout << eValue___getValue() << endl;

   ExpressionValue *eValuePtr1 =
         ______ ExpressionValue(28.0);
   eValuePtr1___setValue(56.0);
   cout << eValuePtr1___getValue() << endl;

   ExpressionValue *eValuePtr2 = ___eValue;
   cout << eValuePtr2___getValue() << endl;

   __________  eValuePtr1;

   return 0;
}
```

```
int myInt = 8;
int *myIntPointer;
myIntPointer = &myInt;
myIntPointer = new int;
int *myOtherIntPointer =
myIntPointer;
*myOtherIntPointer = 7;
int justAnInt = *myIntPointer;
cout << justAnInt << endl;
```

```
int a = 1, b = 2, c = 3;
int *p1 = &a;
int *p2 = &b;
int *p3 = &c;
p1 = p2;
*p1 = *p3;
*p3 = 5;
cout << a << " " << b << " " << c << endl;
```

```
int *anotherIntPtr = new int[10];
for (int i = 0; i < 10; i++) {
    anotherIntPtr[i] = i;
}

anotherIntPtr = &(anotherIntPtr[2]);
cout << *anotherIntPtr << endl;

for (int i = 0; i < 8; i++) {
    cout << anotherIntPtr[i] << endl;
}

int justAnInt = 7;
anotherIntPtr = &justAnInt;
cout << *anotherIntPtr << “ ”;
cout << anotherIntPtr[0] << endl;
```