CSCI 200: Foundational Programming Concepts & Design

Exam 2 Review

1. What is printed?

```
void my_func( int &x, int y ) {
    x = 52;
    y = 7;
}

int main() {
    int x = 0;
    int y = 0;
    my_func( x, y );
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    return 0;
}
```

1. What is printed?

```
void my_func( int &x, int y ) {
    x = 52;
    y = 7;
}

int main() {
    int x = 0;
    int y = 0;
    my_func( x, y );
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    return 0;
}
```

x = 52

y = 0

2. String

Write a function called string_append that receives a string as input and outputs a string.

The function needs to return a string that appends to the parameter the text

" is a super coder."

2. String

```
string string_append( const string STR ) {
  return STR + " is a super coder.";
}
```

3. Code

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp
#include <iostream>
using namespace std;

#include "Gnome.h"

int main() {
  Gnome a( 10, 25 );
  cout << a._value1 << " "
       << a._value2 << endl;
  return 0;
}
```

3. Questions

a) What message would the compiler display?

b) Correctly rewrite the line of code to correct the error.

c) What is the purpose of const in the two member functions?

d) What is Gnome() and why doesn't it have a return type?

3. Questions

a) Private member _value1 not accessible

b) Correctly rewrite the line of code to correct the error.

c) What is the purpose of const in the two member functions?

d) What is Gnome() and why doesn't it have a return type?

3. Questions

a) Private member _value1 not accessible

b) cout << a.getValue1() << " " << a.getValue2();

c) What is the purpose of const in the two member functions?

d) What is Gnome() and why doesn't it have a return type?

3. Questions

a) Private member _value1 not accessible

b) cout << a.getValue1() << " " << a.getValue2();

c) The callee will not be changed by the function

d) What is Gnome() and why doesn't it have a return type?

3. Questions

a) Private member _value1 not accessible

b) cout << a.getValue1() << " " << a.getValue2();

c) The callee will not be changed by the function

d) A constructor

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;
  Gnome g2();
  g1._value1 = 52;
  int _value1;
  _value1 = g1.getValue1();
  Gnome g3 = g1;
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();
  g1._value1 = 52;
  int _value1;
  _value1 = g1.getValue1();
  Gnome g3 = g1;
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO but YES
  g1._value1 = 52;
  int _value1;
  _value1 = g1.getValue1();
  Gnome g3 = g1;
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;
  _value1 = g1.getValue1();
  Gnome g3 = g1;
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();
  Gnome g3 = g1;
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();          // YES
  Gnome g3 = g1;
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();          // YES
  Gnome g3 = g1;                     // YES
  g3.g2();
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();          // YES
  Gnome g3 = g1;                     // YES
  g3.g2();                           // NO
  cout << _value1 << endl;
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();          // YES
  Gnome g3 = g1;                     // YES
  g3.g2();                           // NO
  cout << _value1 << endl;           // YES
  cout << _value2 << endl;
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();          // YES
  Gnome g3 = g1;                     // YES
  g3.g2();                           // NO
  cout << _value1 << endl;           // YES
  cout << _value2 << endl;           // NO
  return 0;
}
```

4. What is legal?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
private:
  int _value1;
  int _value2;
};
```

```
// main.cpp - assume appropriate headers
int main() {
  Gnome g1;                          // YES
  Gnome g2();                        // NO
  g1._value1 = 52;                   // NO
  int _value1;                       // YES
  _value1 = g1.getValue1();          // YES
  Gnome g3 = g1;                     // YES
  g3.g2();                           // NO
  cout << _value1 << endl;           // YES
  cout << _value2 << endl;           // NO
  return 0;                          // YES
}
```

5. Short Answer

- Suppose you have developed a class called MyClass with private data members x and y of type int.
  a) Write the function header for this class's default constructor.
  b) Write the function implementation for this class's default constructor that sets x and y to 0.

5. Short Answer

- Suppose you have developed a class called MyClass with private data members x and y of type int.
  a) MyClass();

  b) Write the function implementation for this class's default constructor that sets x and y to 0.

5. Short Answer

- Suppose you have developed a class called MyClass with private data members x and y of type int.

a) MyClass();

b)
```
MyClass::MyClass() {
    _x = 0;
    _y = 0;
}
```

6. Functions

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) What is the name of the function?

b) Is this function a member function? If yes, to what class?

6. Functions

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) What is the name of the function?
   doSomething()

b) Is this function a member function? If yes, to what class?

6. Functions

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) What is the name of the function?
   doSomething()

b) Is this function a member function? If yes, to what class?
   Yes, Circle

7. Functions cont.

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) What does the first Circle represent?

b) What does the second Circle represent?

c) What does the third Circle represent?

d) What does the const represent?

7. Functions cont.

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) The return type

b) What does the second Circle represent?

c) What does the third Circle represent?

d) What does the const represent?

7. Functions cont.

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) The return type

b) Class the function belongs to

c) What does the third Circle represent?

d) What does the const represent?

7. Functions cont.

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) The return type

b) Class the function belongs to

c) The parameter type

d) What does the const represent?

7. Functions cont.

```
Circle Circle::doSomething( const Circle &C ) {
  // does something here
}
```

a) The return type
b) Class the function belongs to
c) The parameter type
d) The parameter cannot be modified by the function

8. Constructors

- Which of the following are valid constructors? Justify the issue if one exists.
  a) `BankAccount::BankAccount() const`
  b) `BankAccount::BankAccount( double balance )`
  c) `void BankAccount::BankAccount()`
  d) `BankAccount::BankAccount( const string &acct, double balance )`

8. Constructors

- Which of the following are valid constructors? Justify the issue if one exists.

a) `BankAccount::BankAccount() const`

b) `BankAccount::BankAccount( double balance )`

c) `void BankAccount::BankAccount()`

d) `BankAccount::BankAccount( const string &acct, double balance )`

9. Member Functions

- Which of the following are valid member functions implementation headers? Justify the issue if one exists.

a) `double HotDog::getPrice() const`

b) `Triangle::calculateArea()`

c) `Buffalo Buffalo::buffalo( Buffalo buffalo )`

d) `void Dog::fetchBall`

e) `double AlarmClock::ring( float )`

9. Member Functions

- Which of the following are valid member functions implementation headers? Justify the issue if one exists.

a) `double HotDog::getPrice() const`

b) `Triangle::calculateArea()`

c) `Buffalo Buffalo::buffalo( Buffalo buffalo )`

d) `void Dog::fetchBall`

e) `double AlarmClock::ring( float )`

10. What is printed?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
  int diff();
  int diff( const Gnome &G );
private:
  int _value1;
  int _value2;
};
```

```
int Gnome::diff() {
  return _value2 - _value1;
}
int Gnome::diff( const Gnome &G ) {
  return this->_value2 - G._value1;
}
```

```
int main() {
  Gnome a( 10, 25 ), b( 5, 20 );
  cout << a.diff() << " "
       << a.diff( b ) << endl;
  return 0;
}
```

10. What is printed?

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, int );
  int getValue1() const;
  int getValue2() const;
  int diff();
  int diff( const Gnome &G );
private:
  int _value1;
  int _value2;
};
```

```
int Gnome::diff() {
  return _value2 - _value1;
}
int Gnome::diff( const Gnome &G ) {
  return this->_value2 - G._value1;
}
```

15 20

```
int main() {
  Gnome a( 10, 25 ), b( 5, 20 );
  cout << a.diff() << " "
       << a.diff( b ) << endl;
  return 0;
}
```

11. Army of Gnomes!

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, string );
  int getValue1() const;
  string getName() const;
private:
  int _value1;
  string _name;
};
```

- Declare a vector of Gnomes. Then add two Gnomes:
  - harry with value 35
  - sally with value 38

```
int main() {



}
```

11. Army of Gnomes!

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, string );
  int getValue1() const;
  string getName() const;
private:
  int _value1;
  string _name;
};
```

- Declare a vector of Gnomes. Then add two Gnomes:
  - harry with value 35
  - sally with value 38

```
int main() {
  vector<Gnome> army;
  Gnome harry(35, "harry");
  Gnome sally(38, "sally");
  army.insert(0, harry);
  army.insert(1, sally);
}
```

11. Army of Gnomes!

```
// Gnome.h
class Gnome {
public:
  Gnome();
  Gnome( int, string );
  int getValue1() const;
  string getName() const;
private:
  int _value1;
  string _name;
};
```

- Declare a vector of Gnomes. Then add two Gnomes:
  - harry with value 35
  - sally with value 38

```
int main() {
  vector<Gnome> army;
  army.insert(0, Gnome(35, "harry"));
  army.insert(1, Gnome(38, "sally"));


}
```

12. Composition

```
class Chair { // in Chair.h
public:
  Chair();
  Chair( int, int, int, double );
  // all getters and setters
private:
  int _height, _width, _depth;
  double _price;
};
class Table { // in Table.h
public:
  Table();
  Table( int, int, int, double );
  // all getters and setters
private:
  int _height, _width, _depth;
  double _price;
};
```

- Write a .h file to define a new class DiningSet. DiningSet has two chairs and one table, a bool on whether the set is sold, and a getPrice() function.

12. Composition

```
// in DiningSet.h

#include "Chair.h"
#include "Table.h"

class DiningSet{
public:
  DiningSet();
  double getPrice();
  // all getters and setters
private:
  Chair _chair1, _chair2;
  Table _table;
  bool _sold;
};
```

13. Composition

a) Write the function implementation of the Chair's default constructor. Use 10.0 for the price and 1 for the height, width, and depth.

b) Write the implementation of getPrice() for your DiningSet class. getPrice() is equal to the sum of the table and chairs price.

13. Composition

a) Chair::Chair() {

```
  _price = 10;
  _height = _width = _length = 1;
}
```

13. Composition

b)
```
double DiningSet::getPrice() {
    return _chair1.getPrice()
           + _chair2.getPrice()
           + _table.getPrice();
}
```

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. What is the final value of a & b?
2. What do c, d, e, f, g point to?
3. f is what type of copy of c?
4. g is what type of copy of c?
5. Which of Lines 15, 16, 17 will result in an error? Why? What is the error?

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. What do c, d, e, f, g point to?
3. f is what type of copy of c?
4. g is what type of copy of c?
5. Which of Lines 15, 16, 17 will result in an error? Why? What is the error?

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. c → address of a (value of 3)
   d → address of b (value of 9)
   e → value of 7
   f → address of a (value of 3)
   g → value of 2
3. f is what type of copy of c?
4. g is what type of copy of c?
5. Which of Lines 15, 16, 17 will result in an error? Why? What is the error?

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. c → address of a (value of 3)
   d → address of b (value of 9)
   e → value of 7
   f → address of a (value of 3)
   g → value of 2
3. Shallow Copy
4. g is what type of copy of c?
5. Which of Lines 15, 16, 17 will result in an error? Why? What is the error?

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. c → address of a (value of 3)
   d → address of b (value of 9)
   e → value of 7
   f → address of a (value of 3)
   g → value of 2
3. Shallow Copy
4. Deep Copy
5. Which of Lines 15, 16, 17 will result in an error?
   Why? What is the error?

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. c → address of a (value of 3)
   d → address of b (value of 9)
   e → value of 7
   f → address of a (value of 3)
   g → value of 2
3. Shallow Copy
4. Deep Copy
5. Line 16
   Why? What is the error?

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. c → address of a (value of 3)
   d → address of b (value of 9)
   e → value of 7
   f → address of a (value of 3)
   g → value of 2
3. Shallow Copy
4. Deep Copy
5. Line 16
   Cannot delete memory allocated on the stack

14. Pointers

```
01 int a = 5;
02 int b = 6;
03 int *c = &a;
04 int *d = &b;
05 int *e = new int(7);
06 int *f = new int;
07 int *g = new int;
08 f = c;
09 *g = *c;
10 a = 8;
11 *d = 9;
12 *f = 1;
13 *g = 2;
14 *c = 3;
15 delete e;
16 delete f;
17 delete g;
```

1. a == 3, b == 9
2. c → address of a (value of 3)
   d → address of b (value of 9)
   e → value of 7
   f → address of a (value of 3)
   g → value of 2
3. Shallow Copy
4. Deep Copy
5. Line 16
   Cannot delete memory allocated on the stack
   Memory Leak from Lines 06 & 08

15. Pointers Part 2

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  *pX = 4;
}

void bar(int*& pY) {
  *pY = 5;
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

1. What is the output?
2. Sketch out the memory usage.

15. Pointers Part 2

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  *pX = 4;
}

void bar(int*& pY) {
  *pY = 5;
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 4
3 - 5
```

2. Sketch out the memory usage.

15. Pointers Part 2

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  *pX = 4;
}

void bar(int*& pY) {
  *pY = 5;
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 4
3 - 5
```

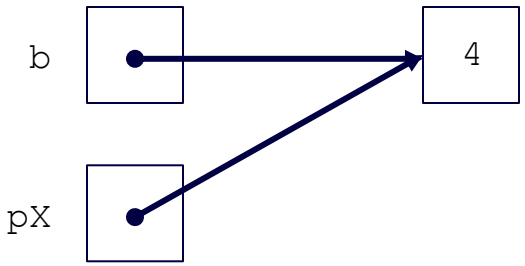
2. Sketch out the memory usage.

15. Pointers Part 2

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  *pX = 4;
}

void bar(int*& pY) {
  *pY = 5;
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 4
3 - 5
```

2. Sketch out the memory usage.



15. Pointers Part 2

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  *pX = 4;
}

void bar(int*& pY) {
  *pY = 5;
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 4
3 - 5
```

2. Sketch out the memory usage.

16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

1. What is the output?
2. Sketch out the memory usage.

16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 2
3 - 5
```

2. Sketch out the memory usage.

16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 2
3 - 5
```

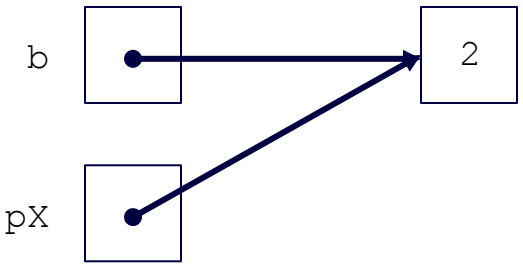
2. Sketch out the memory usage.

16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 2
3 - 5
```

2. Sketch out the memory usage.

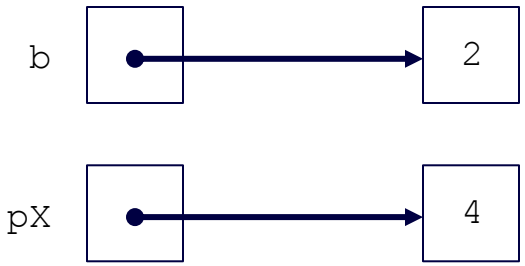


16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 2
3 - 5
```

2. Sketch out the memory usage.

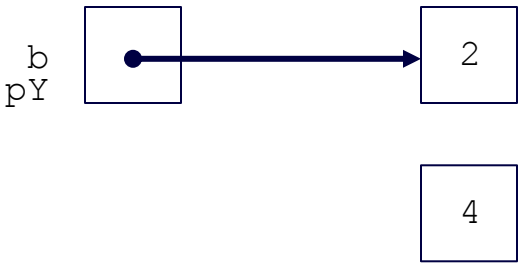


16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 2
3 - 5
```

2. Sketch out the memory usage.

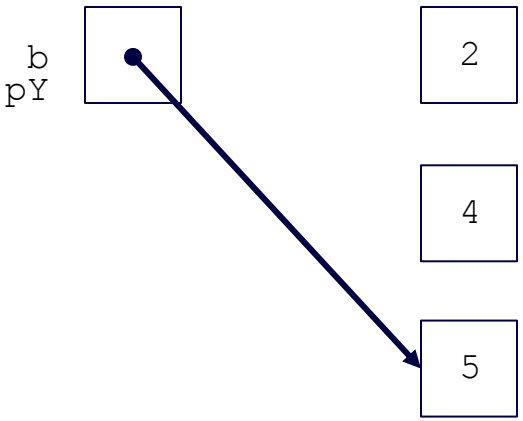


16. Pointers Part 3

```
#include <iostream>
using namespace std;

void foo(int* pX) {
  pX = new int(4);
}

void bar(int*& pY) {
  pY = new int(5);
}

int main() {
  int *b = new int(2);
  cout << "1 - " << *b << endl;
  foo(b);
  cout << "2 - " << *b << endl;
  bar(b);
  cout << "3 - " << *b << endl;
  return 0;
}
```

```
1 - 2
2 - 2
3 - 5
```

2. Sketch out the memory usage.



17. Analysis

- What is the run time of the following block of code?

```
int matches = 0;
string line1, line2;
getline(cin, line1);
getline(cin, line2);
int shorterLine = min(line1.length(), line2.length());
for(int i = 0; i < shorterLine; i++) {
  if(line1.at(i) == line2.at(i)) {
    matches++;
  }
}
if(line1.length() == line2.length() && line1.length() == matches) {
  cout << "Lines are equal" << endl;
} else {
  cout << "Lines are not equal" << endl;
}
```

17. Analysis

- What is the run time of the following block of code? O(n)

```
int matches = 0;
string line1, line2;
getline(cin, line1);
getline(cin, line2);
int shorterLine = min(line1.length(), line2.length());
for(int i = 0; i < shorterLine; i++) {
  if(line1.at(i) == line2.at(i)) {
    matches++;
  }
}
if(line1.length() == line2.length() && line1.length() == matches) {
  cout << "Lines are equal" << endl;
} else {
  cout << "Lines are not equal" << endl;
}
```

18. Analysis 2

- What is the run time of the following block of code?

```
int matches = 0;
string line1, line2;
getline(cin, line1);
getline(cin, line2);
for(int i = 0; i < line1.length(); i++) {
  for(int j = i; j < line2.length(); j++) {
    if(i == j) {
      if(line1.at(i) == line2.at(j)) {
        matches++;
      }
    }
  }
}
if(line1.length() == line2.length() && line1.length() == matches) {
  cout << "Lines are equal" << endl;
} else {
  cout << "Lines are not equal" << endl;
}
```

18. Analysis 2

- What is the run time of the following block of code? $O(n^2)$

```
int matches = 0;
string line1, line2;
getline(cin, line1);
getline(cin, line2);
for(int i = 0; i < line1.length(); i++) {
  for(int j = i; j < line2.length(); j++) {
    if(i == j) {
      if(line1.at(i) == line2.at(j)) {
        matches++;
      }
    }
  }
}
if(line1.length() == line2.length() && line1.length() == matches) {
  cout << "Lines are equal" << endl;
} else {
  cout << "Lines are not equal" << endl;
}
```

19. Analysis 3

- What is the run time of the following block of code?

```
int matches = 0;
string line1, line2;
getline(cin, line1);
getline(cin, line2);
for(int i = 0; i < line1.length(); i++) {
  for(int j = i; j < line2.length(); j++) {
    if(i == j) {
      if(line1.at(i) == line2.at(j)) {
        matches++;
      }
      break;
    }
  }
}
if(line1.length() == line2.length() && line1.length() == matches) {
  cout << "Lines are equal" << endl;
} else {
  cout << "Lines are not equal" << endl;
}
```

19. Analysis 3

- What is the run time of the following block of code? $O(n)$

```
int matches = 0;
string line1, line2;
getline(cin, line1);
getline(cin, line2);
for(int i = 0; i < line1.length(); i++) {
  for(int j = i; j < line2.length(); j++) {
    if(i == j) {
      if(line1.at(i) == line2.at(j)) {
        matches++;
      }
      break;
    }
  }
}
if(line1.length() == line2.length() && line1.length() == matches) {
  cout << "Lines are equal" << endl;
} else {
  cout << "Lines are not equal" << endl;
}
```

20. Analysis 4

- Of Questions 17, 18, 19:
  - Which have the best performance?
  - The worst?

20. Analysis 4

- Of Questions 17, 18, 19:
  - Q17, Q19 – $O(n)$
  - The worst?

20. Analysis 4

- Of Questions 17, 18, 19:
  - Q17, Q19 – $O(n)$
  - Q18 – $O(n^2)$

21. The Big 3

- What are the Big 3?
- What is the Rule of 3?
- Why should we follow the Rule of 3? What can occur if we don't?
- How do the Big 3 relate to shallow/deep copies? What is the difference between the two?

21. The Big 3

- Copy Constructor, Copy Assignment Operator, Destructor
- What is the Rule of 3?
- Why should we follow the Rule of 3? What can occur if we don't?
- How do the Big 3 relate to shallow/deep copies? What is the difference between the two?

21. The Big 3

- Copy Constructor, Copy Assignment Operator, Destructor
- If we implement 1, then implement all 3
- Why should we follow the Rule of 3? What can occur if we don't?
- How do the Big 3 relate to shallow/deep copies? What is the difference between the two?

21. The Big 3

- Copy Constructor, Copy Assignment Operator, Destructor
- If we implement 1, then implement all 3
- Explicit resource management. Prevent memory leaks, dangling pointers
- How do the Big 3 relate to shallow/deep copies? What is the difference between the two?

21. The Big 3

- Copy Constructor, Copy Assignment Operator, Destructor
- If we implement 1, then implement all 3
- Explicit resource management. Prevent memory leaks, dangling pointers
- Default implementation is shallow copy, only copy references.
  - Deep copy duplicates values in new memory

22. Programming Paradigms

- What is the difference between Procedural Programming and Object-Oriented Programming?

- Write an example block of code that illustrates each style in use.

22. Programming Paradigms

- POP – manipulate external state via series of function calls
  OOP – manipulate internal state encapsulated within object via series of function calls

- Write an example block of code that illustrates each style in use.

22. Programming Paradigms

- POP – manipulate external state via series of function calls
  OOP – manipulate internal state encapsulated within object via series of function calls

- POP - sort(list)
  OOP – list.sort()

23. File I/O

- Given a file named “xc.txt” with the following data

```
n
x1 x2 x3 … xn
```

  - Where the first integer in the file, `n`, states how many integers will follow in the file (`n` will be at least 1)

- Write a program to read in all the integers and print out the largest & smallest integer.

23. File I/O

```
#include <fstream>
#include <iostream>
using namespace std;

int main() {
  ifstream fin("xc.txt");
  if( !fin ) { return -1; }

  int n;
  fin >> n;

  int tempVal, min, max;
  fin >> tempVal;
  min = max = tempVal;
// continues
```

```
// continues
   for(int i = 1; i < n; i++) {
    fin >> tempVal;
    if(tempVal < min) min = tempVal;
    if(tempVal > max) max = tempVal;
   }

  fin.close();

  cout << "min: " << min << endl;
  cout << "max: " << max << endl;

  return 0;
}
```

Questions?