1. [extra credit] Who created C?

   a. Gosling
   b. Kildall
   c. Ritchie
   d. Stroustrup
   e. Thompson
   f. van Rossum
   g. Wirth
   h. Wall

2. [3 pts]Given

```
int theAnswer = 17;
const int* p = &theAnswer;  // Line A
```

Which of the following is true?

   a. ~~Line A does not compile~~
   b. `*p = 42;  // Does not compile`
   c.
```
int another = 42;
p = &another;  // Does not compile
```
   d. None of the above

3. [3 pts]Given:

```
class SomeClass {
public:
    SomeClass() {}
private:
    int a;
    Foo b;
    Bar* c;
};

int main() {
    SomeClass thing;
}.
```

How are the member variables of `thing` defined in `main` being initialized?

- If the variable is not initialized answer NOT.
- If it is initialized then indicate *how* it is initialized,
  either what value (such as zero or NULL) is stored in it, or what function is used to initialize it.
- Assume Foo and Bar have all necessary methods and constructors.

Member Variable	Initialized how?
a	initialized to some garbage number supplied by your system.
b	Foo's default constructor called
c	NOT

Questions 4–12 are four points each.

4. Given a class called `Thing` and the code

```
Thing thingOne, thingTwo;
```

What function **call** is the following line equivalent to?

```
Thing thingTwo = thingOne;
```

a. `operator=(thingTwo, thingOne)`

b. `thingTwo.operator=(thingOne)`

c. `ostream& Thing::operator=(const Thing& rhs);`

d. Neither a. nor b. because it is using the `Thing` copy constructor.

e. Neither a. nor b. because the operator has to be overridden as a friend

f. Either (a) or (b), depending on how the programmer chose to implement the operator.

g. None of the above

-4

5. Given:

```
class BaseClass {
public:
   void foo() const;
};
```

We want to be sure that all classes that inherit from BaseClass to override the method `foo()`
Modify the above code to guarantee that.

```
class BaseClass {
public:
   virtual void foo() const;
};
virtual void foo() const = 0;
```

-2

Note: Questions 6 and 7 refer to the classes defined below.

Read carefully.

```
class Animal {
public:
   Animal() {}
   virtual void sleep() {cout << "In Animal sleep()"; }
private:
};

class Cat : public Animal {
public:
   virtual void eat() {cout << "In Cat eat()"; }
   void sleep() {cout << "In Cat sleep()"; }
private:
};
```

6. What would be the result of:

```
int main() {
   Animal* p = new Cat();
   p->eat();
}
```

a. The program runs and prints: `In Cat eat()`

b. The program runs and prints: `In Animal eat()`

c. Runtime error

d. Compilation error because there is no Cat constructor

e. None of the above

7. What would be the result of:

```
int main() {
   Cat cat;
   Animal animal;
   animal = cat;
   animal.sleep();
}
```

a) The program runs and prints: `In Cat sleep()`

b) The program runs and prints: `In Animal sleep()`

c) Runtime error

d) Compilation error because there is no Cat constructor

e) Compilation error because Animal cannot be assigned to Cat.

f) None of the above

8. What is the result of compiling and running the following program?

```
class Shape {
public:
        Shape() { display(); }
        virtual void display() { cout << "Shape"; }
};

class Square : public Shape {
public:
        void display() { cout << "Square"; }
};

int main() {
        Square sq;
}
```

-4

a) The program runs and prints: Shape

b) The program runs and prints: Square

c) The program runs and prints: ShapeSquare

d) The program runs and prints: SquareShape

e) Fails to compile because Square constructor is not defined

f) Fails to compile for a reason other than (e)

g) runtime error (or undefined behavior)

h) None of the above

9. A student implements a vector class. Testing his class, he writes some code

```
Vector v;
for (int i = 0; i < 42; ++i) v.push_back(i);
v = 17;   // Huh?
```

0

Looking at his test code, he sees that the last line was a mistake, and is surprised that it compiled without an error.

What is the best way he can fix his Vector class so that the last line will not compile?
(Answer in whatever way you think most appropriate.)

he can make the constructor for the Vector class "explicit", so that it can only be initialized a certain way thus raising a compilation error on the last line.

10. Given:

```
int* data = new int[12];
```

Pick an expression that is equivalent to: data[5]

a) data*5	d) *data+5	g) &(data+5)	j) data+5&
b) &(data*5)	e) *(data+5)	h) (data+5)&	k) data&+5
c) &data+5	f) (data+5)*	i) data+5	

11. What is the result of the following?

```
class Base {};
class Derived : public Base {};

void func(Base& base) { cout << "func(Base)\n"; }
void func(Derived& der) {cout << "func(Derived)\n"; }

void otherFunc(Base& base) {
  func(base);
}

int main() {
  Derived d;
  otherFunc(d);
}
```

a. The program runs and prints: func(Base)

b. The program runs and prints: func(Derived)

c. fails to compile

d. runtime error (or undefined behavior)

e. None of the above

12. Given:

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

class Member {
public:
  Member() {cout << 1;}
};

class Base {
public:
  Base( ) {cout << 2;}
};

class Derived : public Base {
public:
  Derived() {cout << 3;}
  Member member;
};

int main() {
  Derived der;
}
```

-4

What is the output?

a. 123
b. 132
c. 213
d. 231
e. 312
f. 321
g. Fails to compile
h. Runtime error (or undefined behavior)

231

Programming – Blue Book

- Place the answers to the following question in your Blue Book.
- **Comments** are **not** required in the blue book!
  However, if you think they will help us understand your code,
  feel free to add them.
- Read the question *carefully!*

13. [58 pts] You will write a **complete program** defining two classes: Company and Employee.

- A Company
  - has a name and a collection of Employees.
  - can hire Employees
- An Employee
  - has a name
  - can quit his job
- All employees exist on the heap.
- When an employee is hired, the company becomes "responsible" for him.
- Your task is to implement the classes Company and Employee. Yes, the two classes do refer to each other. You must handle that.
- Write a **complete program**.
  - It may be in a single file, but it must be complete, except for
    - main(). You are not required to provide main().
    - Other functions, belonging to Company or Employee, that you are explicitly told you do not have to implement.
- The Company class will have the following:
  - Big 3.
    - As stated, Employees become part of the Company and so their fortunes live and die with the Company. If the Company goes under the Employee does, too... If the Company is cloned, so are the Employees.
    - **Of the Big 3**, you **only have to implement** the **assignment operator**.
  - Output operator ✓
  - Index operator that takes a name and returns the address of the first Employee in the Company with that name.
  - hire method. It is passed the address of an Employee. You may safely assume that the Employee is on the heap.
  - removeEmp method.
    - To save you time, you **do not have to implement this method**. You can use it in your code without implementing it.
    - It is passed the address of an Employee to be removed.
    - It only removes the Employee from the Company's vector.
      It does not modify the Employee or call any functions to do so.
  - Any other functions needed by the program.

[Continued on next page]

- The Employee class will have the following:
  - a constructor that takes the Employee's name
  - a quit method. It takes no arguments. It is called on the Employee when he wishes to quit.
  - Any other methods necessary.

Sample test function:

```
int main() {
  Company comp("hal");
  Employee* fred = new Employee("fred");
  comp.hire(fred);  // The company is now responsible for fred.
  comp.hire(new Employee("mary"));
  Employee* maryPtr = comp["mary"];
  cout << comp << endl;
  maryPtr->quit();
  cout << comp << endl;
}
```

Output for sample:

```
Company: hal; Employees: fred mary.
Company: hal; Employees: fred.
```