

Name: Zaid Aureshi

Poly-Id: 0185649

1. [extra credit] Who created C?

- |                   |               |
|-------------------|---------------|
| a. Gosling        | e. Thompson   |
| b. Kildall        | f. van Rossum |
| <u>c. Ritchie</u> | g. Wirth      |
| d. Stroustrup     | h. Wall       |

[Questions 2 - 10 are worth five points each]

2. Given a class called Thing and the code

```
Thing thingOne, thingTwo;
```

What function call is the following line equivalent to?

```
thingTwo = thingOne;
```

- b.
- 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

3. Given a vector of ints called intVec, write a "ranged for" loop, also sometimes known as a "foreach" loop, to double the values of all the elements in the vector.

0

```
for(int& item : intVec)
{
    item *= 2;
}
```

4. Assume that the class Eeyore has been defined and that the + operator has been overloaded as a non-member function to add two Eeyores. What other function is needed in order to allow the lines below to compile.

5

```
Eeyore e;
e + 5;
```

- Do not overload the + operator again.
- Do not implement this new function. Just give its prototype.

```
Eeyore(int); //Constructor
```

```
Eeyore(int)
```

Prototype: \_\_\_\_\_

5. 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`

6. Given:

```
const int x = 10;
```

Which of the following will compile?

- a. `int* p = &x;`  
b. `int* const q = &x;`  
c. `const int* r = &x;`  
d. All of the above  
e. None of the above

7. What is the output of the following program:

```
class Base {  
public:  
    void foo(int n) { cout << "Base::foo(int)\n"; }  
};  
class Derived: public Base {  
public:  
    void foo(double n) { cout << "Derived::foo(double)\n"; }  
};  
  
int main() {  
    Derived der;  
    der.foo(42);  
}
```

- a. `Base::foo(int)`  
b. `Derived::foo(double)`  
c. The program does not compile  
d. The program does not generate any output.  
e. None of the above

8. Given:

```

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

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

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

int main() {
    Derived der;
}

```

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)

9. Given:

```

class Integer {
public:
    Integer(int n) { val = n; }
private:
    int val;
};

```

What has to be added to the Integer class, so that the following will correctly display "myInt is positive" when the value in myInt is positive:

```

int main() {
    int n;
    cin >> n;
    Integer myInt(n);
    if(myInt) cout << "myInt is positive\n";
}

```

bool operator==( )  
{ if (val > 0) return true; }

Integer::operator bool() const {  
return val > 0; }

Zaid Aureshi

Poly-Id:

0485649

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

```
class Base {};  
class Derived : public Base {  
public:  
    void method() { cout << "method1\n"; }  
};  
class Derived2 : public Base {  
public:  
    void method() { cout << "method2\n"; }  
};  
  
int main() {  
    Base* bp = new Derived();  
    Derived2* d2p = bp;  
    d2p->method();  
}
```

- a. The program compiles and runs, printing "method1"
- b. The program compiles and runs, printing "method2"
- c. The program compiles and runs to completion without printing anything.
- d. The program compiles and crashes when it runs.
- ☒ e. The program does not compile.
- f. None of the above.

David B. Smith

Poly-Id: 0485649

What is the result of the following?

```
class Base {
public:
    virtual void foo() { cout << " - Base::foo()\n"; }
};
class Derived : public Base {
public:
    void foo() { cout << " - Derived::foo()\n"; }
};

void func(Base& arg) {
    cout << "func(Base)";
    arg.foo();
}

void func(Derived& arg) {
    cout << "func(Derived)";
    arg.foo();
}

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

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

- a. The program runs and prints:  
func(Base) - Base::foo()
- b. The program runs and prints:  
func(Derived) - Derived::foo()
- c. The program runs and prints:  
func(Base) - Derived::foo()
- d. The program runs and prints:  
func(Derived) - Base::foo()
- e. The program fails to compile
- f. A runtime error (or undefined behavior)
- g. None of the above



- ...ing - Blue Book*
- the answer 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.
- Includes are not required in the blue book.
  - Read the question carefully!

12. [47 pts] One of the most important jobs in our country is the baker. He makes all those treats that we crave and that provide us with the energy to study for (and write) exams! **You will implement a class** to represent this important national resource.

What do bakers do?

- Make treats! The baker has to create treats on demand, e.g.  
`theBaker.bakes("Twinkie");`
  - Each treat will be created on the heap so it can have a long shelf-life.
  - The Treat class has
    - a constructor that takes a string which is that name of the Treat
    - an output operator that displays the name.
    - Anything else Treats have is a trade secret. (Ok, they support copy control.)
- Deliver the treats to a company, who will in turn package them and sell them (to us!).
  - This requires that he pass the collection off to the bakery company. After handing the collection over, he is back to having nothing. All those treats, and in fact the container itself that held them now belongs to the company. E.g.  
`aCompany.receive(theBaker.delivers());`

Just to keep life entertaining, we will want to support copy control for our baker.

- Deep copy, of course.
- To keep this exam to a reasonable length, you only have to implement the assignment operator. For the other copy control functions, just provide their prototypes in the class and assume someone else will write them for you.

And naturally you should provide a reasonable output operator displaying him and his products. See the sample test code and output on the next page.

And finally, let's have an equality operator. We will consider two bakers to be "equal" if they currently have the same number of treats.

So, what do you have to implement? **Just the Baker class.**

- You don't have to worry about the bakery company or even defining the Treat class. (Well, worry all you like, but you are not writing those classes.)
- You can also assume that Treat provides any needed operators or constructors.
- The baker has:
  - a constructor ✓
  - bakes method ✓
  - delivers method ✓
  - output operator ✓
  - equals operator ✓
  - copy control ✓

anureshi

Poly-Id: 0485649

... so how will you represent all of this? You should be able to work out a good design, but let me "help"

- Since the baker hands over the collection of treats that he has baked, he needs to have a *pointer* to the collection.
- No, don't ask me what sort of collection to use. Use whatever you like. But the baker better be free to bake and store as many treats in your container as needed. We don't know how many that will be.
- And the **collection itself needs to be on the heap**.
- When the baker bakes a treat, if he has no place to put it, either because he just came on the job or because he just delivered his collection to a company, he will first need to get / create another collection. Where? Again, obviously from the heap.
- The treats will later be repackaged and sold and eaten, finally being destroyed in the process. No you don't have to represent all of those steps, but clearly the treats must each exist on the heap so they can have an arbitrarily long shelf-life and can be moved about as needed by all the code that will use your class.

Sample test code:

```
int main() {
    Baker fred("fred");
    cout << fred << endl;
    fred.bakes("Twinkie");
    fred.bakes("Cupcake");
    fred.bakes("Twinkie");
    fred.bakes("Twinkie");
    fred.bakes("Cupcake");
    fred.bakes("Wonderbread");
    cout << fred << endl;

    Baker joe("joe");
    cout << joe << endl;
    joe = fred;
    cout << fred << endl;
    cout << joe << endl;

    // Don't implement Bakery
    Bakery hostess("Hostess");
    hostess.receive(fred.delivers());
    cout << fred << endl;
    cout << joe << endl;
}
```

Resultant output:

```
Baker: fred; No treats :-(
Baker: fred; Twinkie Cupcake Twinkie Twinkie Cupcake Wonderbread.
Baker: joe; No treats :-(
Baker: fred; Twinkie Cupcake Twinkie Twinkie Cupcake Wonderbread.
Baker: fred; Twinkie Cupcake Twinkie Twinkie Cupcake Wonderbread.
Baker: fred; No treats :-(
Baker: fred; Twinkie Cupcake Twinkie Twinkie Cupcake Wonderbread.
```