Name: Urs Evora Poly-Id: 0499140

Short Answer

Note: #includes have been omitted to save space. Assume that if they are needed, they are there.

For multiple choice questions, circle only one answer

1. [extra credit] [4 pts] Who created C++?

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

2. [4 pts] Given:

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

Which of the following is true? (Choose one!)

a. Line A does not compile.

b. *p = 42; // Does not compile

c. int another = 42;
   p = &another; // Does not compile

d. (b) and (c)
   because the Line A above does not compile.

e. (b) and (c)
   but line A above does compile.

f. None of the above

3. [4 pts] Given a vector of ints called intVec, use a "ranged for" loop, also sometimes known as a "foreach" loop, to compute the sum of all the elements in the vector.

```
int counter = 0;
for(int i : intVec) {
    counter += i;
}
```

Name: Urs Evorg Poly-Id: 0499140

Questions 4 and 5 use the following classes:

```
class Pet {
public:
    Pet() {}
    virtual void bar() {cout << "In Pet bar()"; }
};

class Cat : public Pet {
public:
    virtual void eat() {cout << "Cat eating"; }
    virtual void bar() {cout << "In Cat bar()"; }
};
```

4. [4 pts] Given the above classes, what would be the result of:

```
int main() {
    Pet* petPtr = new Cat();
    petPtr->eat();
}
```

a. The program runs and prints: `Cat eating`

b. The program runs and prints: `Pet eating`

c. The program compiles but has a runtime error

d. Compilation error because there is no Cat constructor

e. Compilation error other than (d).

f. None of the above

5. [4 pts] Given the above classes, what would be the result of:

```
int main() {
    Cat felix;
    Pet peeve;
    peeve = heathcliffe;
    peeve.bar();
}
```

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

b. The program runs and prints: `In Pet bar()`

c. Runtime error.

d. Compilation error.

e. None of the above

Name: Urs Erora Poly-Id: 0499140

6. [4 pts] What would be the result of:

```
class Embedded {
public:
      Embedded() { cout << "1"; }
};
class Base {
public:
      Base() { cout << "2"; }
      Embedded e;
};
class Derived : public Base {
public:
      Derived() { cout << "3"; }
};

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

a. 23
b. 32
c. 123
d. 132
e. 213
f. 231
g. 312
h. 321
i. Does not compile
j. None of the above.

7. [4 pts] Given

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

class Derived : public Base {
public:
      Derived() { }
      void display() { cout << "Derived"; }
};

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

For the classes Base and Derived defined above write the code in main() that will call the Base display method on the object der.

```
int main() {
      Derived der;
      // Put your code here
      der.Base::display();

}
```

Name: Urs Erora Poly-Id: 0499140

Programming – Blue Book

- Place the answers to the following questions 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.
- **Do not use iterators!!!** (Sorry for the shouting)

8. [54 pts]
In one company anyone in the company may have a staff. Or not. Yes, even the lowliest intern in the mail room has the possibility of acquiring a staff.

There are some limitations to who can be on your staff. You can't hire yourself. You can't hire your immediate boss. (While it may seem silly to be allow hiring your boss's boss, etc., the company decided that implementing such a test would be expensive.) Oh, you also can't hire someone who already is on someone's staff.

Employees may leave a group by calling their quit method. If someone tries to quit who is not in a group, nothing happens. (We might need to send them to the company's shrink, but that's not your program's responsibility.)

BTW, lots of people in the company have the same names. It's a strange place.

Below is a test program and the resulting output. Note that if we use your class definition, the output should match, except as regarding the order of an employee's staff, which can printed in any order.

Implement only the Employee class. No includes required.

Test Program

```
int main() {
    Employee groucho("Groucho");
    Employee harpo("Harpo");
    Employee chico("Chico");
    Employee zeppo("Zeppo");

    //Status
    groucho.display();
    harpo.display();
    chico.display();
    zeppo.display();
    cout << "=====\n";

    groucho.hire(harpo); // return true
    harpo.hire(groucho); // return false
    groucho.hire(zeppo); // return true
    groucho.hire(chico); // return true

    //Status
    groucho.display();
    harpo.display();
    chico.display();
    zeppo.display();
    cout << "=====\n";

    harpo.quit();
    harpo.hire(groucho);

    //Status
    groucho.display();
    harpo.display();
    chico.display();
    zeppo.display();
    cout << "=====\n";
}
```

Sample Output

```
Name: Groucho; Boss: none; Staff: none.
Name: Harpo; Boss: none; Staff: none.
Name: Chico; Boss: none; Staff: none.
Name: Zeppo; Boss: none; Staff: none.
=====
Name: Groucho; Boss: none; Staff: Harpo Zeppo Chico.
Name: Harpo; Boss: Groucho; Staff: none.
Name: Chico; Boss: Groucho; Staff: none.
Name: Zeppo; Boss: Groucho; Staff: none.
=====
Name: Groucho; Boss: Harpo; Staff: Chico Zeppo.
Name: Harpo; Boss: none; Staff: Groucho.
Name: Chico; Boss: Groucho; Staff: none.
Name: Zeppo; Boss: Groucho; Staff: none.
=====
```

9. [22 pts]

Given the type `Thing`:

```
struct Thing {
  string foo;
  int bar;
};
```

write the two functions,

- **fill**: fills a vector with data from a file. The lines of the file each have a string and an integer, to be used for the `foo` and the `bar` fields respectively.
- **release**: frees up all of the memory that was allocated, zeroing the size of the vector.

Below is an example program in which `fill` and `release` are called from main.

```
int main() {
  ifstream ifs("things.txt");
  vector<Thing*> things;
  fill(ifs, things);      // Implement this function
  release(things);        // Implement this function
}
```

8.
```
class Employee {
public:
    Employee(const string& newName)
    : name(newName) {}

    bool hire(Employee& newStaff) {
        if(boss == &newStaff || this == &newStaff){
            return false;
        }
        else if(checkInStaff(newStaff)){ return false;}
        else if(newStaff.hired) { return false;}

        else{
            newStaff.quit();
            staff.push_back(&newStaff);
            newStaff.boss(this);
            newStaff.hired(true);
        }
    }
    bool checkInStaff(const Employee& newStaff) const{
        bool inStaff = false;
        for(Employee* e : staff){
            if(e == &newStaff){
                inStaff = true;
            }
        }
        return inStaff;
```
boss? -5

Why?

-1

```
}
void quit() {
    if( boss != NULL ) {
        for( size_t i = 0; i < boss->staff.size(); i++ ) {
            if( boss->staff[i] == this ) {
                boss->staff[i] == boss->staff[boss->staff
                boss->staff.pop_back();
                hired = false;
            }
        }
    }
}

void display() const {
    cout << "Name: " << name << "; Boss: ";
    if( boss == NULL ) {
    cout << "none; Staff: ";
    }
    else { cout << boss->name << "; Staff: "; }
    if( staff.size() == 0 ) { cout << "none." << endl; }
    else {
        for( Employee* e : staff ) {
            cout << e->name << " ";
        }
    cout << "." << endl;
    }
```

return?

size() - 1];