CS20A Assignment 3

NASHIR JANMOHAMED

TOTAL POINTS

27.25 / 30.5

QUESTION 1

1 Problem 1 3 / 3

✓ - **0 pts** Correct

- **1 pts** implicit versions
- **1.5 pts** Shallow Copy
- **0.5 pts** Pointers and memory (or resources), but not necessarily created by class.
- **1 pts** Pointers, but no dynamic mem
- **1.5 pts** Dynamic Memory Allocation
- **3 pts** Incomplete

QUESTION 2

2 Problem 2 3 / 3

✓ - **0 pts** Correct

- **0.5 pts** Reuse: Explanation
- **0.5 pts** Reuse: Example
- **0.5 pts** Extend: Explanation
- **0.5 pts** Extend: Example
- **0.5 pts** Specialize: Explanation
- **0.5 pts** Specialize: Example
- **1.5 pts** No clear discussion on the 3 advantages.
- **2 pts** Providing an description rather than discussing advantages

QUESTION 3

3 Problem 3 3 / 3

✓ - **0 pts** Correct

- **1 pts** Discussed constructor calls, but did not indicate order of object construction.
- **1.5 pts** Incorrect construction order
- **1.5 pts** incorrect destruction order

QUESTION 4

4 Problem 4 3 / 3

✓ - **0 pts** Correct

- **1 pts** Engine in inheritance hierarchy
- **0.5 pts** Engine as Prius member

QUESTION 5

5 Problem 5 2 / 3

- **0 pts** Correct
- **+ 1.5 pts** Incomplete argument
- **- 3 pts** Incorrect.
- **0 pts** Click here to replace this description.

- 1 Point adjustment

Chopping occurs with the "correct" assignment, this case is much worse.

QUESTION 6

6 Problem 6 2 / 3

- **0 pts** Correct
- **+ 1.5 pts** Polymprphism, virtual functions
- **+ 3 pts** Incorrect

- 1 Point adjustment

Only if it is defined as virtual

QUESTION 7

7 Problem 7 1 / 1

✓ - **0 pts** Correct

- **0.5 pts** Incomplete
- **1 pts** Incorrect

QUESTION 8

8 Problem 8 1.5 / 2.5

- **0 pts** Correct

✓ - **0.5 pts** **delete p1 or temp; not both**

- **0.5 pts** for () delete p[] p2[]
- **0.5 pts** for () delete p3[];
- **0.5 pts** delete [] p3;

✓ - **0.5 pts** **delete[] p4;**

- **0.5 pts** Missing for loop logic

QUESTION 9

9 3 / 3

✓ - **0 pts** **Correct**

- **0.5 pts** int **rows, ...
- **0.5 pts** ..., *col1, *col2, *col3;
- **0.5 pts** new int*[3];
- **0.5 pts** *(rows + ...
- **0.5 pts** &col
- **0.5 pts** *(*(rows + i) + j) ...

QUESTION 10

10 1.75 / 2

- **0 pts** Correct
- **0.5 pts** sp1=&d1;
- **0.5 pts** sp2=&sp1; (**sp2).set(d2.get());
- **0.5 pts** sp3=&sp2; **sp3=&d3;
- **0.5 pts** address of d3.
- **0.25 pts** Syntax errors

✓ - **0.25 pts** **Logical errors**

QUESTION 11

11 2 / 2

✓ - **0 pts** **Correct**

- **0.5 pts** NV: 6 Apples
- **0.5 pts** NV: 3 Oranges
- **0.5 pts** V: 7 Apples
- **0 pts** V: 2 Oranges
- **2 pts** Incorrect

QUESTION 12

12 2 / 2

✓ - **0 pts** **Correct**

- **2 pts** Incorrect
- **1 pts** Shallow copy to b
- **1 pts** Badge Copy Constructor
- **0.5 pts** Explanation unclear/incomplete

gradescope

Santa Monica College
Spring 2018

CS 20A: Data Structures with C++

Assignment 3

Name: Nashir Jaumohamed

ID: ~~304325140~~ 1325605

Concepts:

Problem 1:
If you do not explicitly define a copy constructor or assignment operator for your objects, (1) what happens when you attempt to make copies or assign one object to another? (2) In what cases must you define a copy constructor and assignment operator?

(1) The compiler will make a "shallow" copy. In other words, all the data will be directly assigned from the passed in object. All values will be copied and any pointer member variables will point to the same location.

(2) Anytime we have dynamically allocated memory we must explicitly define a copy constructor & assignment operator.

* If we use the default copy constructor or assignment operator when we have dynamically allocated member variables, the address of the passed in objects member variables is copied directly, which will almost inevitably lead to runtime errors.

Problem 2:
What are the primary advantages of inheritance, provide a small programing example with two classes illustrating each point.

There are 3 primary advantages of inheritance:

(1) Reuse - using code from base classes in subclasses

(2) Extend - adding new behavior or data to a subclass

(3) Specialization - redefining an existing behavior (from base class) with a new behavior

Example

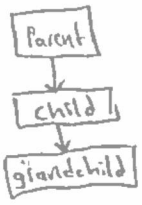
```
class Shape {
public:
    virtual double getArea()=0;
    void output() {
        cout << "this is a shape!\n";
    }
};
```

```
class Square : public Shape {
public:
    Square (double x): m_side(x){ };
    virtual double getArea() {
        return (m_side * m_side);
    }
private:
    double m_side;
};
```

Name: Nashir Janmohamed **ID:** 1325603

Problem 3:

Suppose Child is a class derived from the class Parent, and the class Grandchild is a class derived from the class Child. This question is concerned with the constructors and destructors for the three classes Parent, Child, and Grandchild. When a constructor for the class Grandchild is invoked, what constructors are invoked and in what order? When the destructor for the class Grandchild is invoked, what destructors are invoked and in what order?

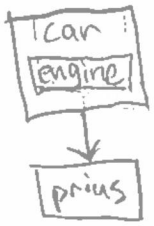


1) When an object of type Grandchild is created, three constructors are invoked. First is the Parent constructor, then the Child constructor, and finally the Grandchild constructor.

2) When the object is destroyed the object destructors are called in the reverse order – first the Grandchild destructor is called, then the Child destructor is called, and finally, the Parent destructor is invoked.

Problem 4:

Suppose you were tasked with developing a program that incorporated the following three objects: Car, Prius, and Engine. How would you organize these objects' relationship with one another? Why?



I would create the Engine class first, then create the Car class and give it an Engine. Finally, because the Prius, needs an engine & most (if not all) of the features of the car, I would make it inherit from the Car class, and add whatever additional functionality I might need.

Problem 5:

Why can't we assign a base class object to a derived class variable?

We can't do this because we lose the functionality of the derived class variable and risk running into runtime errors. The variable is "chopped" meaning all derived data members are no longer used.

Problem 6:

Suppose the base class and the derived class each have a member function with the same signature. When you have a pointer to a base class object and call a function member through the pointer, discuss what determines which function is actually called—the base class member function or the derived-class function.

C++ defines a <u>vtable</u> when we define a variable of a class. The entries of the vtable point to the functions that should be used for the given variable. When we have pointers, the computer looks at what type of object the pointer points to, and calls the appropriate function. When you have a pointer to a base class object, the base-class member function is called.

Name: Nashir Janmohamed **ID:** 1325609

Program Problems

For the following programming problems make an attempt **without** using your computer this can be done on scratch paper if you prefer. After the initial attempt you may program the problems to check you work. Your final answer should be written on the assignment.

Problem 7:

This program is supposed to write 1 4 9 16 25 36 49 64 81 100 , but it probably does not. What is the problem with this program? (We're not asking you to propose a fix to the problem.)

```
int* computeSquares(int& n) {
      int arr[10];
      n = 10;
      for (int k = 0; k < n; k++)
            arr[k] = (k + 1) * (k + 1);
      return arr;
}

void f() {
      int junk[100];
      for (int k = 0; k < 100; k++)
            junk[k] = 123400000 + k;
}

int main() {
      int m;
      int* ptr = computeSquares(m);
      f();
      for (int i = 0; i < m; i++)
            cout << ptr[i] << ' ';
}
```

arr is not dynamically allocated. this could be a problem because it is deleted once you leave the scope of computeSquares(); which result in garbage values being printed.

Name: Nashir Jannohamed **ID:** 1325605

Problem 8:

Write delete statements that correctly delete the following dynamically allocated entities.

```
int *p1 = new int[10];
```

delete [] p1;

```
int *p2[15];
for (int i = 0; i < 15; i++)
      p2[i] = new int[5];
int **p3 = new int*[5];
for (int i = 0; i < 5; i++)
      p3[i] = new int;
int *p4 = new int;
int *temp = p4;
p4 = p1;
p1 = temp;
```

for (int i=0; i<15; i++)
delete [] p2[i];
for (int i=0; i<5; i++)
delete p3[i];
delete [] p3
delete p4;

Problem 9:

Consider the code fragment below. It is supposed to construct a 3x4 (3 rows 4 columns) 2d array of integers and set each value to zero. However, as given it does not. Add the proper dereferences (*) or references (&) to make this code work properly:

```
int **rows,  *col1,  *col2,  *col3;

rows   = new   int*[3];      // Create 3 pointers to columns
col1   = new   int[4];       // Create first row with 4 elements
col2   = new   int[4];       // Create second row with 4 elements
col3   = new   int[4];       // Create third row with 4 elements

*(  rows + 0 ) = &col1[0];   // Point to first row
*(  rows + 1 ) = &col2[0];   // Point to second row
*(  rows + 2 ) = &col3[0];   // Point to third row

for (int i = 0; i<3; i++)
      for (int j = 0; j<3; j++)
            *(  *(   rows + i) + j) = 0;// rows[i][j] = 0;
```

Name: Nashir Janmohamed **ID:** 1325605

Problem 10:

For this problem, you will be asked to write some code to accomplish a particular task given the code fragment below. Each task may depend on the tasks that came before it. Your code must be syntactically correct.

```
class S {
public:
      S(int init) :m_num(init) {}
      S() :m_num(0) {}              // ↓no parentheses needed
      void set(int num) {m_num = num);}
      int get() { return m_num; }
private:
      int num; //should be m_num
};

S d1, d2(4), d3(-15);
S *sp1, **sp2, ***sp3;
```

Set sp1 to point to d1.

sp1 = &d1;

Using sp2 change the value of *num* in d1 to the value of *num* in d2 (you may not use d1).

sp2 = &sp1;
**sp2 = d2;

Using sp3 make sp1 point to d3 (you may not use sp1).

sp3 = &sp2; // *sp3 = &sp1
**sp3 = &d3; // **sp3 = sp1 //&d1

What does the following code output? `cout<< **&*sp3;` If it is a value, state the value, if it is an address state the name of the variable to which the address belongs.

It prints the value pointed to by sp1, which is the address of d3.

sp3; //address of sp2
*sp3; //contents of sp2 == &sp1
**sp3; // contents of sp1 == &d3

Name: Nashir Janmohamed **ID:** 1325605

Problem 11:

Consider the following program:

```
class A {
public:
      A() :m_msg("Apple") {}
      A(string msg) : m_msg(msg) {}
      virtual ~A() { message(); }
      void message() const {
            cout << m_msg << endl;
      }
private:
      string m_msg; = Apple
};
```

```
class B :public A {
public:
      B() :A("Orange") {}
      B(string msg) : A(msg), m_a(msg) {}
      void message() const {
            m_a.message();
      }
private:
      A m_a;
};
```

```
int main() {
      A *b1 = new B;
      B *b2 = new B;
      A *b3 = new B("Apple");
      b1->message();
      b2->message();
      (*b3).message();
      delete b1;
      delete b2;
      delete b3;
}
```

How many times will you see the word Apple in the output? 6

How about Orange? 3

Now make A's message() virtual, i.e.,

```
virtual void message() const;
```

How many times will you see the word Apple in the output? 7

How about Orange? 2

Name: Nishir Junmohamed **ID:** 1325605

Problem 12:

Consider the following program and generated output:

```
class Security {
public:
   Security(int id)
       :m_id(id), m_badge(id % 10) {}

   ~Security() {
       cout << "Security::~Security: "
              << m_id << endl;
   }
   // Get badge reference
   Badge & badge() {
       cout << "Security::badge: Ret ref "
              << endl;
       return m_badge;
   }
   // Get badge value
   Badge  badgeV() {
       cout << "Security::badge: Ret val "
              << endl;
       return m_badge;
   }
private:
   int m_id; = 1
   Badge m_badge; = 1
};
```

```
class Badge {
public:
    // Constructor
    Badge(int num) :m_num(num) {
       m_stuff = new int[6];
       for (int i = 0; i < 6; i++) {
              m_stuff[i] = num;
       }
    }
    // Destructor
    ~Badge() {
       cout << "Badge::~Badge: " << m_num
              << endl;
       delete[] m_stuff;
    }

    void setNum(int num) {
       m_num = num;
       for (int i = 0; i < 6; i++) {
              m_stuff[i] = num;
       }
    }

    void print() {
       cout << "Badge Num: ";
       for (int i = 0; i < 6; i++) {
              cout <<m_stuff[i];
       }
       cout << endl;
    }
private:
    int m_num; = 1
    int *m_stuff; = [1,1,1,1,1,1]
};
```

```
1 int main() {
2    Security s(11);
3    s.badge().print();
4    if (true) {
5        Badge b = s.badge();
         b.setNum(2);
         b.print();

         s.badge().print();

         cout << "Main::Leaving if:"
                << endl;
     }
     cout << "Main::Leaving main:" << endl;
}
```

```
Output:
 Security::badge: Ret ref
 Badge Num: 111111
 Security::badge: Ret ref
 Badge Num: 222222
 Security::badge: Ret ref
 Badge Num: 222222
 Main::Leaving if:
 Badge::~Badge: 2
 Main::Leaving main:
 Security::~Security: 11
 Badge::~Badge: 1
```

Name: Nashir Janmohamed **ID:** 1325605

Question on the next page:

This program runs fine until the very end where we experience a runtime crash. Using a debugger we discover that there is an exception when calling the destructor for Badges at the line `delete[] m_stuff;`. What is causing this crash how would you improve the Badge class to prevent this from occurring?

We didn't create a copy constructor, so when we initialize "b" in line 5, it makes a shallow copy of all the data in s.badge, and when it goes out of scope of the if statement, the destructor deletes the dynamically allocated m_stuff (which both "m_badge" and "b" point to).

Example Copy Constructor

```
Badge (const Badge &src){
    m_num = src.m_num
    m_stuff = new int[6];
    for (int i=0; i<6; i++)
        m_stuff[i] = src.m_stuff[i];
}
```