Name: Stewart Dulaney ID: 1545566

Santa Monica College

CS20A Data Structures with C++

Midterm Exam

Fall 2018

Closed Book, Closed Notes, No Electronic Devices.

Please write your name once on each sheet.
Use scratch paper for intermediate work.
Clearly indicate your final answer.

Good Luck!

Name: Stewart Dulaney

True/False: Circle either True or False.

1.True (False) Suppose you have an int pointer called ptr pointing to an integer array. C++ understands *(ptr) + 2 to mean "move two integers down and retrieve that memory location". int* ptr

2.True (False) Not defining a constructor for your class will result in a compiler error.

3.True (False) Declaring a pointer to an object will call the constructor for that object.

4.True (False) A shallow copy is sufficient for handling dynamically allocated member variables.

5.True (False) structs are less useful than classes because you cannot create a constructor for structs.

6.True (False) If you declare an array of N objects the constructor for that object is called once to create that array.

7.True (False) The following class definition for A requires the entire class definition for B in order to compile.

```
class A {
public:
     void foo( B b);
private:
     B* pb;
};
```

8.(True) False The new keyword requests memory to be allocated by the operating system and returns an address to the allocated memory.

9.(True) False Eliminating errors at the pencil and paper stage makes it much easier to produce a correct program in the later steps of the problem-solving process.

10.(True) False Template is a feature of C++ that allows us to write one code for different data types.

11.True (False) When overloading the assignment operator for an object the return value is always a non-reference object type.

12.True (False) Having protected member variables is consistent with Encapsulation.

The principle of encapsulation says implementation details should be hidden from the user, including when the user is a derived class.

Name: Stewart Dulaney

Multiple Choice: Choose the answer that fits best. Write out your letter choice into the provided space and circle your choice.

1. d Abstract Data Types consist of:
   a. Data Structures.
   b. Algorithms.
   c. An Interface.
   (d.) All mentioned.

2. d Suppose the variable s is a pointer to a structure that has a member variable called address. Which of the following is the correct way to access that member variable:
   a. s->address;
   b. (*s).address;
   c. s[0].address;
   (d.) All of the above.
   e. None mentioned.

3. b You must include a class's header file when.
   a. Have the class as a parameter to a function.
   (b.) Use any of that class's member functions.
   c. Declare a pointer or reference to that class.
   d. Have the class as a return type of a function
   e. All mentioned.

4. d What is the lifetime of dynamically allocated data?
   a. In the function it is defined.
   b. Within the main function.
   c. Within the class it is allocated.
   (d.) Until it is deleted.
   e. None mentioned.

5. b A class with at least one pure virtual function is called:
   a. Abstract Data Type.
   (b.) Abstract Base Class.
   c. Polymorphic Template.
   d. Dynamically Allocated.
   e. None mentioned.

Name: Stewart Dulnney

6. d A copy constructor for an object is called:
   - a. Only if you explicitly define one.
   - b. When making an assignment between two existing objects.
   - c. When you want to make a shallow copy of an object.
   - (d.) When creating a new object from an already existing object
   - e. None mentioned

7. c Which keyword can be used in creating Templates?
   - a. class.
   - b. typename.
   - (c.) both class and typename.
   - d. function.
   - e. None mentioned.

8. d What is the output of the following program?

```
#include<iostream>
using namespace std;
class Point {
public:
    Point() { cout << "Constructor called" << endl; }
};

int main() {
   Point t1, *t2;
   return 0;
}
```

   - a. Compiler Error.
   - b. Constructor called
     Constructor called
   - c. 0
   - (d.) Constructor called
   - e. Nothing will print.

Name: Stewart Dulaney

9. e Given the following class definitions, what is the resulting output:

```
class Robot {
public:
     virtual void dance() = 0; // pure virtual
     virtual void sing() = 0; // pure virtual
     void compute() { cout << "Beep "; }
};
class HappyRobot :public Robot {
public:
     virtual void dance() { cout << ":) "; }
     void compute() { cout << "Boop "; }
};
void main() {
     Robot *r = new HappyRobot();
     r->dance();
     r->sing();
     r->compute();
}
```

HappyRobot is an abstract base class

a. :) 0 Boop
b. 0 0 Beep
c. 0 0 Boop
d. :) 0 Beep
e. Compile error.

10. c Including the class definitions above with the one below, what is the resulting output?

```
class EcstaticRobot : public HappyRobot {
public:
     virtual void sing() { cout << ":D "; };
     void compute() { cout << "Woop "; }
};
void main() {
     Robot *r = new EcstaticRobot();
     r->dance();
     r->sing();
     r->compute();
}
```

:) :D Beep

a. :) :D Boop
b. :) :D Woop
c. :) :D Beep
d. 0 0 Beep
e. Compile error.

Name: Stewart Dulaney

Short Answer:

1. This following program is supposed print 30 20 10, but it does not. Find all the bugs and rewrite the corrected version of the program. Make as minimal amount of changes as possible while maintaining the overall structure of the program; cout<<"30 20 10"; is not a correct fix.

```
int main(){
     int arr[3] = { 5, 10, 15 };
     int* ptr = arr;

     *ptr = 10;           // set arr[0] to 10
     *ptr + 1 = 20;       // set arr[1] to 20
     ptr += 2;
     ptr[0] = 30;         // set arr[2] to 30

     while (ptr >= arr) {
          ptr--;
          cout << ' ' << ptr;    // print values
     }
     cout << endl;

}
```

```
#include <iostream>
using namespace std;
int main () {
    int arr[3] = {5,10,15};
    int * ptr = arr;

    *ptr = 10;
    *(ptr+1) = 20;
    ptr += 2;
    ptr[0] = 30;

    while ( ptr >= arr ) {
        cout << *ptr;
        if ( ptr == arr ) {
            break;
        }
        cout << " ";
        ptr--;
    }
    cout << endl;
}
```

Name: Stewart Dulaney

2. Suppose you have the two objects defined below:

```
class Book {  // private:
      string title;
      string author;
};

class Person {
public:
      Person(string name, int age, int nbooks)
            :m_num_books(nbooks), m_name(name), m_age(age)
      { m_books = new Book[m_num_books]; }

      string getName() { return m_name; };
      int    getAge() { return m_age; }
      void   cheer() { cout << "I like lemonade!" << endl; }

      ...

private:
      string m_name;
      int m_age;
      Book * m_books;
      int m_num_books;
};
```

a. If you have the following main, discuss what happens and if it results in any logical errors. If there are any issues you just need to discuss them, not fix them.

```
int main() {
      Book b1;
      b1.author = "Neil Gaiman";
      b1.title = "American Gods";
      Book b2 = b1;    // copy constructor
}
```

The class Book's member variables author and title are private (by default) so attempting to access them from main results in a compile error. If lines 2 and 3 set Book's members using setter functions instead, the copy constructor call in the last line would work fine b/c it would perform a shallow copy, which is sufficient b/c the class Book has no dynamically allocated variables.

Name: Stewart Dulaney

b. If you have the following main, discuss what happens and if it results in any logical errors. If there are any issues you just need to discuss them, not fix them.

```
int main() {
      Person p1("Mr. Smith", 9999, 3), p2("Neo", 28, 1);
      for (int i = 0; i < 10000000; i++)
            p2 = p1;  // assignment operator
}
```

This will cause unintended aliasing of dynamic memory because the Person class contains a dynamically allocated array of Books and there is no assignment operator implemented, so the default assignment operator just does a shallow copy.

c. Implement the destructor for Person. Assume this done outside the class definition.

```
Person::~Person() {

    delete [] m_books;

}
```

d. Implement the copy constructor for Person. Assume this done outside the class definition.

```
Person::Person(const Person &other) {
    m_name = other.m_name;
    m_age = other.m_age;
    m_num_books = other.m_num_books;
    m_books = new Book[m_num_books];
    for (int i=0; i<m_num_books; i++) {
        m_books[i] = other.m_books[i];
    }
}
```

Name: Stewart Dulaney

e. Overload the assignment operator for Person. Assume this done outside the class definition.

```
Person& Person::operator=(const Person &other) {
    if (this == &other) { return *this; }
    delete [] m_books;
    m_name = other.m_name;
    m_age = other.m_age;
    m_num_books = other.m_num_books;
    m_books = new Book[m_num_books];
    for (int i=0; i < m_num_books; i++) {
        m_books[i] = other.m_books[i];
    }
    return *this;
}
```

3. Assuming all the issues above are addressed correctly, in addition to the earlier two classes, suppose you have a Student.

```
class Student : public Person {
public:
        Student(string name, int age, int nbooks, int id, int nclasses)
                :m_id(id), m_num_classes(nclasses)
        { m_classlist = new string[m_num_classes]; }
        int getID() { return m_id; }
        void cheer() { cout << "Got any grapes?" << endl; }

        ...
private:
        int m_id;
        string *m_classlist;
        int m_num_classes;
};
```

a. There is a syntax issue that occurs when we try to declare an instance of Student, show how would you fix this?

A compile error occurs b/c C++ attempts to implicitly call the default constructor for Person which is not defined. To fix this we need to explicitly call the Person constructor w/ the required parameters in the initializer list like below:

```
Student(...): Person(name, age, nbooks), m_id(id), m_num_classes(nclasses)
{
    ...
}
```

Name: Stewart Dulaney

b. Implement the destructor for Student. Assume this done outside the class definition.

```
Student :: ~ Student () {

    delete [] m_classlist;


}
```

The destructors for Person and Student should also be declared virtual in their class definitions so C++ knows to call the derived destruct when implementing polymorphism.

c. Implement the copy constructor for Student. Assume this done outside the class definition.

```
Student:: Student ( const Student &other ) : Person (other) {
    m_id = other. m_id;
    m_num_classes = other. m_num_classes;
    m_classlist = new string [m_num_classes];
    for ( int i=0; i < m_num_classes ; i++ ) {
        m_classlist [i] = other. m_classlist [i];
    }


}
```

d. Overload the assignment operator for Student. Assume this done outside the class definition.

```
Student& Student:: operator = ( const Student &other ) {
    if ( this == &other ) { return *this; }
    delete [] m_classlist;
    Person:: operator = (other);
    m_id = other. m_id;
    m_num_classes = other. m_num_classes;
    m_classlist = new string [m_num_classes];
    for ( int i=0; i < m_num_classes; i++ ) {
        m_classlist [i] = other. m_classlist [i];
    }

    return *this;
}
```

Name: Stewart Dulaney

e. What is printed with the following main:

```
int main() {
      Student *Kvoth = new Student("Kvoth", 15, 1, 00013, 8);
      Kvoth->cheer();
}
```

Output:

Got any grapes?

f. Consider the following main where we intend on using polymorphism of Person to behave like a Student. However, as Person is currently defined this does not result in that behavior, further we find that our process takes up increasing amounts of memory as it runs. What changes to Person do you have to make to fix these issues.

```
int main() {
      for (int i = 0; i < 10000000; i++) {
            Person *Kote = new Student("Kvoth", 15, 1, 00013, 8);
            Kote->cheer();
      }
}
```

- The function cheer() should be declared virtual in Person as well as ^ in Student (for clarity) so that C++ knows to use Student's definition of cheer().

- The Person destructor should be declared virtual and the Student destructor should also be declared virtual so that C++ knows to call the Student destructor as well as the Person destructor. This should fix the memory leak.

Name: Stewart Dulaney

4. What is the output of the following program?

```
class Paw {
public:
   Paw() {cout<<"P ";}
   ~Paw() {cout<<"~P ";}
};
```

```
class FluffyButt {
public:
   FluffyButt()  {cout<<"F ";}
   ~FluffyButt() {cout<<"~F ";}
};
```

```
class Doge {
public: // call Paw c'tor 4 times
   Doge() {cout<<"D ";}
   ~Doge() {cout<<"~D ";} // call Paw d'tor 4 times
   virtual void run(){cout<<"R ";}
   void bark() {cout<<"B ";}
private:
   Paw m_paws[4];
};
```

```
class Corgi :public Doge {
public: // call Doge c'tor // call FluffyButt c'tor
   Corgi() {cout<<"C ";}
   ~Corgi() {cout<<"~C ";} // call FluffyButt d'tor // call Doge d'tor
   virtual void run() {cout<<"r ";}
   void bark() {cout<<"b ";}
private:
   FluffyButt m_butt;
};
```

```
   void main() {
      Corgi Ein;
      cout << endl << "---" << endl;
      Ein.bark();
      Ein.run();
      cout << endl << "---" << endl;
      Doge *Pluto = &Ein;
      cout << endl << "---" << endl;
      Pluto->bark();
      Pluto->run();
      cout << endl;
      cout << endl << "---" << endl;
}
```

Output:

P P P P D F C
---
b r

// blank line

B r
// blank line

~C ~F ~D ~P ~P ~P ~P

Name: Stewart Dulaney

5. Rewrite the following function so that it can operate on arrays of any data type, not just integers. You may assume that all operations and comparisons are defined for all data types that we would pass into this function:

```
int cloneArrayReturnMax(int* arr1, int size1, int* arr2, int size2) {
    int end = size1;
    if (end > size2) end = size2;

    for (int i = 0; i < end; i++)
        arr1[i] = arr2[i];

    int max = arr1[size1 - 1];

    for (int i = size1 - 2; i >= 0; i--)
        if (max < arr1[i]) max = arr1[i];

    return max;
}
```

```
template <typename DataType>
DataType cloneArrayReturnMax(DataType* arr1, int size1, DataType* arr2, int size2) {
    int end = size1;
    if (end > size2) end = size2;

    for (int i=0; i<end; i++)
        arr1[i] = arr2[i];

    DataType max = arr1[size1-1];

    for (int i = size1-2; i>=0; i--)
        if (max < arr1[i]) max = arr1[i];

    return max;
}
```