University of Michigan – Dearborn

Department of Computer & Information Science

CIS 200

Midterm Examination February 24, 2021

Time: 150 minutes

Professor: J. Shen

Total Marks: 60 Closed Book

Name: ____Demetrius Johnson_________________________

Email: _________meech@umich.edu________

Question 1

a)

Unix controls file access by two methods: Unix separates users who access a file in three categories as a hierarchical structure: users, groups, and other. Users have the ownership of a particular node, groups consist of groups names that have a certain level of permissions, and other refers to all other users who try to use the file.

The second method is through file access level. There are three file permissions: Read, Write, and Execute (R-W-X). So, for each file on a UNIX system it specifies user, group, and other, and for each of those three categories it tells which of the permissions each type has.

For example: FileNameExample.pdf RWX-RW----X. If there is a dash in place of the permission, it means that the user group or other does not have that permission. So from the example above, the User has Read Write and Execute permissions, the Groups has only Read and Write permission, and other can only Execute.\

b)

```
$ date | mail –s “Date” shen@umich.edu
```

the following command uses a pipe; thus for this scenario the output for the date command will be emailed to shen@umich.edu with the subject of the email being “Date”.

```
$ man sort > foo.txt
```

The first command uses the Manual command which can be applied to any command to display a manual on how to use the command. In this case, ‘man’ is being applied to the ‘sort’ command for inquiry; the > will send the output from the command to a file called foo.txt.

c)

“mkdir” command allows a user to create a new directory (folder/node) in the current working directory and give it a name: for example → “mkdir newFolder” will create a folder called “newFolder” in the current working directory.

“mv” command will move a file or folder to another location specified.

d)

There are many types of UML, it is a non-proprietary modeling language; it stands for “Unified Modeling Language” for that reason, because it is a “language” that can be created and used to model any type of abstract structure. All models are “unified” in that the language describes the spectrum of how something works. It could be a highway system, a compute program, a website, a fire station; basically anything can be unified and modeled as some form of a language. You might considered a Road Map a UML of lots of routes, the names of the routes, and the locations along the route.

For example, a user-case diagram is a UML of the form that matches user-cases/options to another part of a user-interaction system. This could, for instance, when a user is purchasing an item, the chart would show all of the possibilities that a user could do to interact with a website and its paths.

Another example for a UML could be a Class Diagram. These are another very useful UML for software developers, particularly those who program using object-oriented programming such as what we use for this class: C++. A class diagram is an extremely useful tool when a program that needs to be developed will involve a lot of inheritance in order to help keep track. For instance, if you have a large category called “fruit”, the “fruit” class would need to be segmented into many other classes, and class at the next level would likely share/require the attributes to the path of classes above it; thus a class diagram would help model such a large and potentially complex inheritance program.

e)

The major stages of the Software Development Cycle include: Define the problem, Model the solution (using a UML), write the code, debug/test cycle, release versions, and maintenance and updates.

Two types of testing methods include releasing early versions to a limited user set, or doing the normal debug method.

Questions 2

a)

```
struct Employee{
int employment_id;
float salary;
bool marriage_status;
string last_name;
};
```

b)

```
int main(){
Employee x{1234, 34000.00, true, Maxwell}; //I believe this is how you quickly initialize a struct
Employee y = x;

Return 0;
}
```

c)

```
Employee z[2];
z[0]. employment_id = 1235;
z[0]. salary = 35000.00;
z[0]. marriage_status = false;
z[0]. last_name = Fox;

z[1]. employment_id = 1236;
z[1]. salary = 36000.00;
z[1]. marriage_status = true;
z[1]. last_name = Smith;
```

d)

function:

```
void PrintArrayElements(Employee z[]) { /arrays are automatically passed in by reference

        for(int i = 0; i < 2; i++){

        cout << z[i]. employment_id << “ “;
        cout << z[i]. salary << “ “;
        cout << z[i]. marriage_status << “ “;
        cout << z[i]. last_name << endl;

        }

        return;
}
```

now use the function:

```
PrintArrayElements(z);
```

Question 3

a)

out put is:

```
x= 2, y= 3, z= 5.

x= 0, y= 3, z= 10.
```

Scheme: a and c are passed by reference, b is passed by value (copy is made local only to the function thus if a value is passed in at the b parameter it is not affected in MAIN).

b)

first printarray output: 5 10 15

second printarray output: 2 4

Question 4

a)

```
int main(){
#include <fstream>
Grade studentOne(A);
studentOne.print();

ss outputFile(ios::out, “output.txt”);
outputFile << studentOne.print();

outputFile.close();
return 0;
}
```

b)

D: I and III only.

c)

first I will convert to binary so it is easy to do bitwise operations:

a = 5 = 0101, b = 8 = 1000, c = 3 = 0011.

d = a&b = 0000 = 0.

e = a | c = 0111 = 7.

f = a >> 1 (shift right 1 bit) = 0010 = 2.

g = a << 1 (shift left 1 bit) = 1010 = 10.

Thus output is: d= 0, e=7, f=2, g=10.

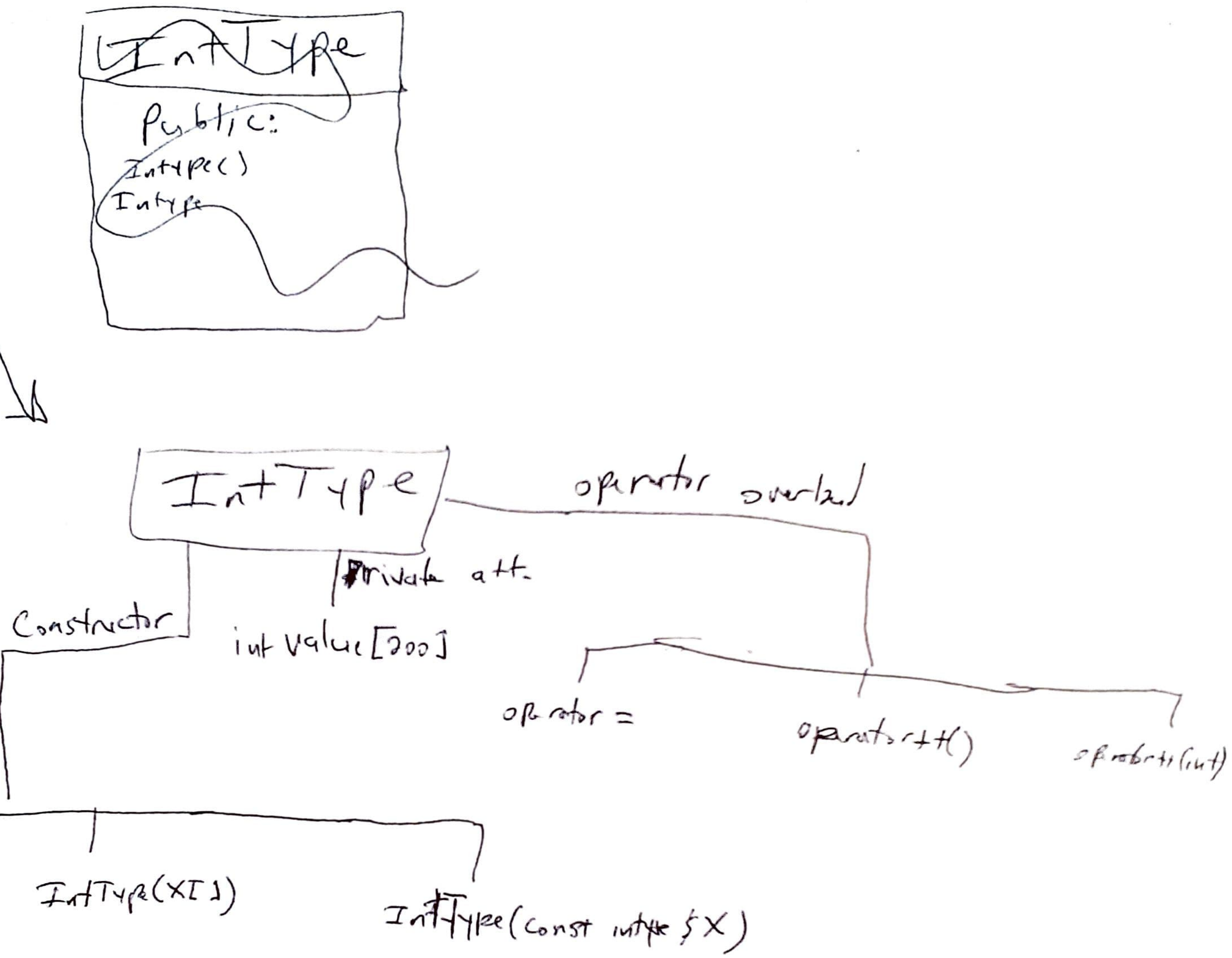
Question 5

```
IntType( ){ //default constructor


    for (int i = 0; i < 200; i++){
    value[i] = 0;
    }
    return;
}
IntType(int x[]){ //parameterized constructor


    for (int i = 0; i < 200; i++){
    value[i] = x[i];
    }
return;
}

IntType& operator=(IntType &y){ //I like to use this-> operator for clarity with this function


    for (int i = 0; i < 200; i++){
    this->value[i] = y.value[i];
    }

return *this;
}
```

```
IntType(const IntType &x){ //I like to use this-> operator for clarity with this function


    for (int i = 0; i < 200; i++){

    this->value[i] = x.value[i];

    }

return;

}


bool operator ==(IntType &y){

    bool status = true;

    for (int i = 0; i < 200; i++) {

        if (this->value[i] == y.value[i]) { status = true; }

        else { status = false; }

    }

return status;

}


IntType& operator ++(){  // pre-fix ++


    for (int i = 0; i < 200; i++){

    value[i] += 1;

    }


return *this;


}
```

```
IntType operator ++(int){  // post-fix ++

        IntType holder;

        for (int i = 0; i < 200; i++){

        this->value[i] += 1;

        }


return holder;


} //my instincts tell me something is not correct with the function setup given by the question but I tried the best I could for this one
```

UML DIAGRAM:


~~IntType~~
~~Public:~~
~~Intype()~~
~~Intype~~
IntType
operator overload
Private att.
int value[200]
Constructor
operator =
operator++()
operator++(int)
IntType();
IntType(X[])
IntType(const inttype $X)

Question 6

a)

If you want to delete an item from an unsorted list, simply go to the location that needs to be deleted, and shift all elements below that position UP by one position so as to overwrite the deleted position and make an empty space at end of list.

b)

A friend function is an easy way to bypass the private data attribute of a class so that another class can use all of that classes data members (have access to them). Friends are one-way, meaning if class A is a friend of B, it does not mean B is a friend of A; it must be explicitly written in both classes for it to work both ways.

c)

protected and private data members differ primarily in this regard: protected members can be accessed by other class scopes if those class scopes inherit the class to which the protected member belongs. Private attributes are only accessible to the class scope, with the exception of the use of friend classes.

For inheritance, public private and protected types specify how the attributes of a derived class will inherit the base class attributes. For example, if a class inherits using private mode, then that means all public and protected members of the base class will be inherited as private attributes to the derived class.

d)

Software engineering is the use of computer complexity and abstraction in order to solve various problems at various different levels of complexity; C++, Assembly language, etc…

You can use the assert library in order to check values before and after different code segment implementations. This will help the program to halt if the assert() comes back false in order that it can be easy to debug logical errors especially and the location to which it most likely occurred.

e)

```
float avgTwoD_Array(int a[]){
    float sum;
    for int (int i = 0, i < 30; i++){
        for int (int j = 0, j < 40; j++){
            sum += a[i][j];
        }
    }
return (sum / (30+40));
}
```