F19 CS20A Midterm

ALARCON PEDRO ANTONI

TOTAL POINTS

44 / 50

QUESTION 1

1 TF 1 **1 / 1**

✓ **- 0 pts Correct: False**

- 1 pts Incorrect

QUESTION 2

2 TF 2 **1 / 1**

✓ **- 0 pts Correct: False**

- 1 pts Incorrect

QUESTION 3

3 TF 3 **1 / 1**

✓ **- 0 pts Correct: False**

- 1 pts Incorrect

QUESTION 4

4 TF 4 **1 / 1**

✓ **- 0 pts Correct: False**

- 1 pts Incorrect

QUESTION 5

5 TF 5 **1 / 1**

✓ **- 0 pts Correct: False**

- 1 pts Incorrect

QUESTION 6

6 TF 6 **1 / 1**

✓ **- 0 pts Correct: True**

- 1 pts Incorrect

QUESTION 7

7 TF 7 **1 / 1**

✓ **- 0 pts Correct: True**

- 1 pts Incorrect

QUESTION 8

8 TF 8 **1 / 1**

✓ **- 0 pts Correct: True**

- 1 pts Incorrect

QUESTION 9

9 MC 1 **1 / 1**

✓ **- 0 pts Correct: d**

- 1 pts Incorrect

QUESTION 10

10 MC 2 **1 / 1**

✓ **- 0 pts Correct: d**

- 1 pts Incorrect

QUESTION 11

11 MC 3 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 12

12 MC 4 **1 / 1**

✓ **- 0 pts Correct: d**

- 1 pts Incorrect

QUESTION 13

13 MC 5 **1 / 1**

✓ **- 0 pts Correct: b**

- 1 pts Incorrect

QUESTION 14

14 MC 6 **1 / 1**

✓ **- 0 pts Correct: d**

- 1 pts Incorrect

QUESTION 15

15 MC 7 **1 / 1**

✓ - 0 pts Correct: c

- 1 pts Incorrect

QUESTION 16

16 MC 8 1 / 1

✓ - 0 pts Correct: 2

- 1 pts Incorrect

QUESTION 17

17 MC 9 0 / 1

- 0 pts Correct: f

✓ - 1 pts Incorrect

QUESTION 18

18 MC 10 0 / 1

- 0 pts Correct: e

✓ - 1 pts Incorrect

QUESTION 19

19 MC 11 1 / 1

✓ - 0 pts Correct: d

- 1 pts Incorrect

QUESTION 20

20 MC 12 1 / 1

✓ - 0 pts Correct: e

- 1 pts Incorrect

QUESTION 21

SA 1 6 pts

21.1 *(ptr+1)=13; 1 / 1

✓ - 0 pts Correct 99 3 6 17 13 -5

- 1 pts Incorrect

21.2 ptr -= 2; 1 / 1

✓ - 0 pts Correct: No change

- 1 pts Incorrect

21.3 ptr[0] = -1; 1 / 1

✓ - 0 pts Correct 99 -1 6 17 13 -5

- 1 pts Incorrect

21.4 *(arr + 1) = 42; 1 / 1

✓ - 0 pts Correct 99 42 6 17 13 -5

- 1 pts Incorrect

21.5 foo(&arr[0], &arr[5]); 1 / 1

✓ - 0 pts Correct: No change

- 1 pts Incorrect

21.6 bar(&ptr[2], arr+2); 1 / 1

✓ - 0 pts Correct: 99 42 17 6 13 -5

- 1 pts Incorrect

QUESTION 22

SA 2 Malazan 6 pts

22.1 2 / 2

✓ - 0 pts Correct: Key things:

Implicit Default constructor that does nothing.

Implicit Copy Constructor, shallow copy. Which is fine since there no pointer member variables pointing to dynamically allocated memory.

- 0.5 pts Implicit Default Constructor, does nothing
- 0.5 pts Implicit Copy Constructor, shallow copy
- 0.5 pts No pointer members, shallow copy ok
- 2 pts Incorrect

22.2 1.5 / 2.5

- 0 pts Correct: key things I'm looking for there are

Assignment op isn't overloaded, so we have a shallow copy. Which in this case is bad due to dynamic memory aliasing and memory leaking.

No implementation for destructor so dynamic memory isn't released

- 0.5 pts No Assignment Operator overloaded
- 1 pts Shallow Copy of dynamic memory, aliasing and memory leaks

✓ - 0.5 pts No Destructor

✓ - 0.5 pts dynamic memory not released correctly

- **2 pts** Incorrect

22.3 **1.5 / 1.5**

✓ **- 0 pts Correct:**

Malazan::~Malazan(){delete [] m_end;}

- **0.5 pts** Scope resolution
- **0.5 pts** delete [] m_end;
- **0.5 pts** Used a for loop
- **0.5 pts** Syntax
- **1.5 pts** Incorrect

virtual must be placed in class definition

QUESTION 23

SA 3 BridgeBurner 9 pts

23.1 **1 / 1**

✓ **- 0 pts Correct: Explicitly call Malazan Constructor in Init. List**

- **0.5 pts** Partially correct
- **1 pts** Incorrect

23.2 **1 / 1**

✓ **- 0 pts Correct:**

BridgeBurner::~BridgeBurner() {delete [] m_munitions;}

- **0.5 pts** delete [] m_munitions;
- **0.5 pts** Used a for loop
- **0.5 pts** Syntax
- **1 pts** Incorrect

23.3 **1.5 / 2**

- **0 pts** Correct: Note the invocation of Malazan's copy ctor

BridgeBurner::
BridgeBurner(const BridgeBurner &src):Malazan(src) {
____m_num_mun = src.m_num_mun;
____m_munitions = new
MoranthMuntion[m_num_mun];
____for (int i = 0; i < m_num_mun; i++)
_____m_munitions[i] = src.m_munitions[i];
}

- **0.5 pts** BridgeBurner::BridgeBurner(const BridgeBurner &src)
- **0.5 pts** :Malazan(src)
- **0.5 pts** m_munitions = new MoranthMuntion[m_num_mun];
for (int i = 0; i < m_num_mun; i++)
____m_munitions[i] = src.m_munitions[i];

✓ **- 0.5 pts Syntax or logic error**

- **2 pts** Incorrect

23.4 **3 / 3**

✓ **- 0 pts Correct:**

BridgeBurner&
BridgeBurner::operator=(const BridgeBurner &src) {

____if (this == &src) return *this;

____Malazan::operator=(src);

____delete [] m_munitions;

____m_num_mun = src.m_num_mun;
____m_munitions = new
MoranthMuntion[m_num_mun];
____for (int i = 0; i < m_num_mun; i++)
_____m_munitions[i] = src.m_munitions[i];

____return *this;
}

- **0.5 pts** BridgeBurner&
BridgeBurner::operator=(const BridgeBurner &src)
- **0.5 pts** if (this == &src)
- **0.5 pts** return *this;
- **0.5 pts** Malazan::operator=(src);
- **0.5 pts** delete [] m_munitions;
- **0.5 pts** m_num_mun = src.m_num_mun;
m_munitions = new MoranthMuntion[m_num_mun];
for (int i = 0; i < m_num_mun; i++)

____m_munitions[i] = src.m_munitions[i];

- **0.5 pts** Syntax or logic
- **3 pts** Incorrect

23.5 1 / 2

- **0 pts** Correct

virtual void moto()...

virtual ~Malazan()... The problem setup asserts that Malazan has a destructor.

- **1 pts** virtual void moto()

✓ - **1 pts** **virtual ~Malazan()**

- **0.5 pts** Just mentioning destructor without virtual
- **2 pts** Incorrect

QUESTION 24

SA 4 Labyrinth 6 pts

24.1 Construction 0.5 / 1

- **0 pts** Correct: I L H H G
- **0.5 pts** Hair x2

✓ - **0.5 pts** **Construction order**

- **0.5 pts** Pointers do have invoke constructors
- **1 pts** Incorrect

24.2 Jareth.intro(); 1 / 1

✓ - **0 pts** **Correct: You remind me of the babe**

- **1 pts** Incorrect

24.3 Bowie->intro(); 1 / 1

✓ - **0 pts** **Correct: You remind me of the babe**

- **1 pts** Incorrect

24.4 Jareth.chorus(); 0 / 1

- **0 pts** Correct: Dance magic, dance

✓ - **1 pts** **Incorrect**

24.5 Bowie->chorus(); 1 / 1

✓ - **0 pts** **Correct: Jump magic, jump**

- **1 pts** Incorrect

24.6 Destruction 1 / 1

- **0 pts** Correct: ~G ~H ~H ~L ~I

✓ - **0 pts** **In reverse order of construction, with construction being incorrect.**

- **0.5 pts** Destruction order
- **0.5 pts** Pointers do have invoke destructors
- **1 pts** Incorrect

QUESTION 25

25 SA 5 Template 3 / 3

✓ - **0 pts** **Correct**

- **0.5 pts** template<typename Data> or template<class Data>
- **0.5 pts** Data * arr1, ... Data * arr2
- **0.5 pts** Data * merge_new(...
- **0.5 pts** Data * arr3 = new Data[s];
- **0.5 pts** Data v1 = ..., Data v2 = ...
- **1 pts** Incorrect change of from int
- **0.5 pts** Minor Syntax
- **3 pts** Incorrect

Name: Pedro Marcos ID: 1722260

Santa Monica College

CS20A Data Structures with C++

Midterm Exam

Fall 2019

Closed Book, Closed Notes, No Electronic Devices.

**Please write your name on this front page,
then your initials once on each sheet.
Use scratch paper for intermediate work.
Clearly indicate your final answer.**

Good Luck!

Name: PA.

True/False: Circle either True or False. int. *ptr

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".

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 If you declare an array of N objects the constructor for that object is called once to create that array.

6.True / False The new keyword requests memory to be allocated by the operating system and returns an address to the allocated memory.

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

8.True / False When overloading the assignment operator for an object the return value is an object reference type. & This

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.

Name: P. A.

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.

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.

Name: P.A.

8. b What does the following print?

```
int mystery(int m, int n) {

   if (m == n)
         return m;

   if (m > n)
         return mystery(m-n, n);

   return mystery(m, n-m);
}
int main() {
   cout << mystery(6, 8) << endl;
}
```

a. 1
b. 2
c. 3
d. 6
e. 8
f. none mentioned

9. h. Recursive functions consists of:

a. Base Cases.
b. Simplifying Steps.
c. Magic Functions.
d. Iteration.
e. All of the above
f. a and b.
g. b and c.
h. a and d.

10. d. Suppose you have just the following declaration: int * ptr[4];
Of the possibilities below, the appropriate way to delete ptr is:

I. delete ptr;
II. delete [] ptr;
III. for(int i=0; i<4; i++) delete ptr[i];

a. I only
b. II only
c. III then I
d. III then II
e. none mentioned

Name: P. A.

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

```
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.

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

```
class Robot {
public:
     virtual void dance() = 0;
     virtual void sing() = 0;
     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();
}
```

inherits pure functions

:)

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

Name: P. A.

Short Answer:

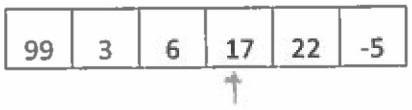
1. Consider the following program, the blocks to the right of main correspond to the memory of the array. Complete the memory after each line of code in main is execute.

```
void foo(int* a, int *b) {
      int* temp = a;
      a = b;
      b = temp;
}

int main() {

   int arr[6]
     = { 99, 3, 6, 17, 22, -5 };
       int *ptr = &arr[3];

   *(ptr + 1)  = 13;


   ptr -= 2;


   ptr[0] = -1;


   *(arr + 1) = 42;


   foo(&arr[0], &arr[5]);


   bar(&ptr[2], arr+2);

}
```

```
void bar(int* a, int *b) {
      int temp = *a;
      *a = *b;
      *b = temp;
}
```

index →

0 1 2 3 4 5



ptr points to

&arr[5] &arr[0]

temp = &arr[0]
&arr[0] = &arr[5]

arr ptr

Foo (

temp

Name: P. A.

2. Suppose you have the two objects defined below:

```
class TragicDemise {
public:
	string get() const { return m_epitaph; }
	void set(string ep) { m_epitaph = ep; }
private:
	string m_epitaph;
};

class Malazan {
public:
	Malazan(string name, int deaths) : m_name(name), m_num_end(deaths) {
		m_end = new TragicDemise[m_num_end];
	}

	void   moto() { cout << "For the Empire!" << endl; }
	//...
private:
	TragicDemise * m_end;
	int m_num_end;
	string m_name;
};
```

a. If you have the following main, discuss what functions are being called, the resulting behavior, and if there are any logical issues.

```
int main() {
	TragicDemise t1;
	t1.set("May the Seven Spirits guard him for all his days");
	TragicDemise t2 = t1;
	//...
}
```

- First, a default constructor is called, though it is not defined in the TragicDemise class, C++ gives us a default one.
- Then, m_epitaph is initialized
- Because we have not defined a copy constructor, C++ provides one for us. In this case, since we did not declare any dynamic variables, we are fine. The default copy constructor does a shallow copy.

Name: P.A.

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() {
        Malazan m1("Toc The Younger", 2);
        for (int i = 0; i < 100; i++) {
                Malazan m2("Toc Anaster", 3);
                m2 = m1;
        }
        //...
}
```

This main attempts to copy the attempt to use the assignment operator even though one hasn't been defined in the Malazan class. This will perform a shallow copy which will create aliases since we have a dynamic variable m_end that is defined in this class

Malazan

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

virtual

```
Malazan::~Malazan() {
    delete [] m_end;
}
```

Name: P. A.

3. In addition to the earlier two classes, suppose you have the following classes. Assume all the issues above are addressed correctly and the relevant functions have been implemented.

```
class MoranthMuntion {
public:
        string getType() const {return m_type;}
        void set(string t) { m_type = t; }
        int getAcc() const { return m_acc; }
        void setAcc(int acc) { m_acc = acc; }
private:
        string m_type;
        int m_acc;

};

class BridgeBurner : public Malazan {
public:
        BridgeBurner(string name, int num_mun) : m_num_mun(num_mun) {
                m_munitions = new MoranthMuntion[m_num_mun];
        }
        void   moto() { cout << "First In, Last Out!" << endl; }
        //...

private:
        MoranthMuntion * m_munitions;
        int m_num_mun;
};
```

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

Issue: No default constructor is defined in Malazan, thus we need to initialize Malazan's member variables in constructor for BridgeBurner.

```
BridgeBurner(string name, int num_mun): Malazan(name, num_mun) {
        m_num_mun = num_mun;
        m_munitions = New MoranthMuntion[m_num_mun];
}
```

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

```
BridgeBurner::~BridgeBurner() {
        delete [] m_munitions;
}
```

Name: P.A.

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

```
BridgeBurner :: BridgeBurner( const BridgeBurner &other ){
    MalazAN :: MalazAN( other );  // invoke MalazAN's copy const
    m_num_mun = other.m_num_mun;
    m_munitions = new MoranthMunition[m_num_mun];
    for (int i=0; i < m_num_mun; i++) {
        m_munitions[i] = other.m_munitions[i];
    }
}
```

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

```
BridgeBurner & BridgeBurner :: operator =(const BridgeBurner &other) {
    if ( this == &other ) return *this;
    MalazAN :: operator =(other); // invoke MalazAN's assig. operator
    delete [] m_munitions;
    m_num_mun = other.m_num_mun;
    m_munitions = new MoranthMunition[m_num_mun];
    for (int i=0; i<m_num_mun; i++) {
        m_munition[i] = other.m_munitions[i];
    }
}
```

Name: P.A.

e. Suppose a fellow student is working with the same classes and they indicate that Keeper should be saying, "First In, Last Out!", but says , "For the Empire!" instead. Also, they indicate that as the program is running they notice the memory usage for it increasing. What might they have forgotten to indicate in their class definitions?

```
int main() {
	for (int i = 0; i < 100; i++) {
		Malazan *Keeper = new BridgeBurner("Paran", 1);
		Keeper->moto();
		delete Keeper;
	}
}
```

The student forgot to make the moto() function virtual. This is why the Base class motto() function is being invoked as opposed to the derived class' function.

Name: P. A.

4. What is the output of the following program?

```
class MaliciousIntent {
public:
    MaliciousIntent() { cout << "I "; }
    ~MaliciousIntent() { cout << "~I "; }
};


class LabyrinthDenizen {
public:
    LabyrinthDenizen() { cout << "L "; }
    virtual ~LabyrinthDenizen() { cout << "~L "; }

    void chorus() {
       cout << "Jump magic, jump" << endl;
    }

     virtual void intro() {
        cout << "What babe? " << endl;
    }

private:

    MaliciousIntent m_mi;
};
```

```
class BigHair {
public:
     BigHair() { cout << "H "; }
     ~BigHair() { cout << "~H "; }
};


class GoblinKing :public LabyrinthDenizen {
public:
     GoblinKing() { cout << "G "; }
     ~GoblinKing() { cout << "~G "; }

     void chorus() {
        cout << "Dance magic, dance" << endl;
     }

     virtual void intro() {
        cout<<"You remind me of the babe,"<<endl;
     }

private:

     BigHair m_hair[2];
};
```

```
void main() {
      GoblinKing Jareth;

      LabyrinthDenizen *Bowie = &Jareth;

      cout << endl << "---" << endl;

      Jareth.intro();

      Bowie->intro();

      cout << "---" << endl;

      Jareth.chorus();

      Bowie->chorus();

      cout << "---" << endl;

}
```

Output:

L I G H H

- - -

You remind me of the babe,

You remind me of the babe,

- - -

Jump magic, jump

Jump magic, Jump

- - -

~H ~H ~G ~I ~L

Name: P. A.

5. The following function takes two arrays of integers and merges them together into a dynamically allocated array, then returns the address of the newly allocated array. 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 * merge_new(int * arr1, int n1,
                int * arr2, int n2) {

    int i = 0, j = 0, k = 0;
    int s = n1 + n2;
    int * arr3 = new int[s];

    while (i<n1 && j <n2) {
          int v1 = arr1[i];
          int v2 = arr2[j];

          if (v1 > v2)
                arr3[k++] = v1;
          else
                arr3[k++] = v2;
    }
    while (i < n1)
          arr3[k++] = arr1[i++];

    while (j < n2)
          arr3[k++] = arr2[j++];

    return arr3;
}
```

```
template <typename T>
T *merge_new(T * arr1, int n1,
             T * arr2, int n2) {

int i=0, j=0, k=0;
int s = n1 + n2;
 T * arr3 = new T[s];

while (i<n1 && j<n2) {
   T v1 = arr1[i];
   T v2 = arr2[j];
   if (v1 > v2)
        arr3[k++] = v1;
   else
        arr3[k++] = v2;
}
while (i<n1)
     arr3[k++] = arr1[i++];
while (j<n2)
     arr3[k++] = arr2[j++];

return arr3;
}
```