To get full marks follow these instructions when applicable:

- Design your classes to avoid code duplication (Reuse your code!)

- Place all header information into a header file and the function definitions into an implementation file.

- Your code should safeguard against memory leaks.

- Your module should safeguard against double definition.

- Write your answers in this document and then submit it.

**Part A:**

**struct RAM {**

**int size;**

**};**

Using the structure “RAM” above, complete the following class named Memory . Your class represents a set of RAMS in a Memory and your design meets the following specifications. The class has

two attributes:

* + m\_memory — a pointer to a dynamically allocated array of RAMS.
  + m\_size — the number of RAMS in the Memory.
* a constructor with one parameter for number of RAMS. This constructor dynamically allocates an array of RAMS to the number RAMS received through the parameter and keeps its address in the m\_memory pointer.
* An overload the += operator for a int as the right operand. This will increase the size of each RAM in the Memory by the amount of the int operand.
* An overloaded << operator so Memory can be printed using cout.
* If invalid data is received you will set the object to a safe empty state where applicable.

**Part B:**

Create an **Interface** named **Employee**. This class has no data members and includes two functions: a modifier named **setData()** and a query named **displayData()**.

**Part C:**

Derive a class named **Doctor** from your **Employee** class, with the following properties/member functions:

• A char array that holds the specialization (The length is unknown).

• An integer that holds an employee identifier.

• A public constructor that takes two arguments, an integer and the address of an unmodifiable char array. These arguments should be used to initialize the values of the doctor’s specialization and identifier.

• A public **setData()** member function. This function asks the user for the doctor’s identifier and specialization. For this function only, you may assume that the user will only enter up to 100 characters.

• A public **displayData()** query that displays the doctor’s specialization and identifier.

• Make sure a **Doctor** can be assigned to another **Doctor** but cannot be copied.

• Make sure there is no memory leak after a **Doctor** object goes out of scope.

**Part D:**

Consider the following function, which calculates and returns the multiplication of the set of values pointed to by x.

**double mult(const double\* x, int n)**

**{**

**double mult = 1.0;**

**for (int i = 0; i < n; i++)**

**mult \*= x[i];**

**return mult;**

**}**

Write a function template that extends this definition to any fundamental type.

**Part E:**

The following program contains **five** different errors that would prevent it from compiling or result in undefined behaviour. Identify them by line number, explain briefly the error, and write a fix for it in the space provided. Each properly identified and fixed error is worth three marks. Assume that this code, once corrected, will compile using the GNU g++ compiler on matrix.

1. #include <iostream>

2. using namespace std;

4. class WEB222 {

5. int \*ids;

6. int size;

7. public:

8. WEB222(const int \*ids\_, int n) {

9. ids = new int[n];

10. ids = ids\_;

11. size = n;

12. }

13. ~WEB222(int n) {

14. cout << "deleting all " << n << " ids." << endl;

15. delete[] ids;

16. }

17. void display(){

18. for(int i = 0; i < size; i++)

19. cout << "id: " << ids[size] << endl;

20. }

21.};

22. void prn(const WEB222& spp){

23. spp.display();

24. }

25. int main() {

26. int ids[3] = {23135, 52134, 67112};

27. WEB222 sjj(ids, 3), sbb;

28. prn(sjj);

29. sbb->display();

30. return 0;

31. }

**ERROR 1:** Line number: \_\_\_\_\_\_\_\_\_

Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fix:

**ERROR 2:** Line number: \_\_\_\_\_\_\_\_\_

Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fix:

**ERROR 3:** Line number: \_\_\_\_\_\_\_\_\_

Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fix:

**ERROR 4:** Line number: \_\_\_\_\_\_\_\_\_

Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fix:

**ERROR 5:** Line number: \_\_\_\_\_\_\_\_\_

Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fix:

**ERROR 6:** Line number: \_\_\_\_\_\_\_\_\_

Problem: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Fix:

**Part F:**

Using your own words, briefly answer the following questions:

1. Explain the difference between declaring the data members as private vs public vs protected.
2. Explain what memory leak is and how to prevent it.
3. Explain shallow and deep copying in classes. Identify the type of the class that is suitable or each
4. Explain the concept of dynamic function binding in the context of inheritance. In your explanation, you should mention some class names and a member function name to demonstrate your understanding. In brief, use some C++ code in your explanation.
5. In a few words, explain the following:

*“In a hierarchy, good design codes the destructor in a base class as virtual, even if no class is currently derived from that base class.”*

1. When implementing a binary operator overload (such as +=), what is the significance of having it be it member operator versus a helper operator?
2. With the concept of class hierarchies and abstract base classes in mind, how can the use of an array of pointers improve the ability to access different portions of the hierarchy? Explain your reasons.

**Part G:**

**Consider the following code. Answer questions 1 to 4.**

#include <iostream>

#include <cstring>

using namespace std;

const int NT = 10;

const int NC = 2;

class Chocolate {

char type[NT + 1];

public:

Chocolate(const char\* t = nullptr) {

if (t) {

strncpy(type, t, NT);

type[NT] = '\0';

}

else {

type[0] = '\0';

}

cout << "C";

}

~Chocolate() {

cout << "~" << type << endl;

}

void display() const {

cout << type << endl;

}

};

class Box {

Chocolate ch[NC];

int nc;

public:

Box() {

nc = 0;

cout << "B";

}

Box& operator+=(const Chocolate& c) {

if (nc < NC) {

ch[nc] = c;

cout << "+";

ch[nc++].display();

}

return \*this;

}

Box(const Box& b) {

nc = b.nc;

for (int i = 0; i < nc; i++)

ch[i] = b.ch[i];

cout << "E" << endl;

}

~Box() {

cout << "~B" << endl;

}

void display() const {

for (int i = 0; i < nc; i++)

ch[i].display();

cout << endl;

}

};

void show(const Box b) {

b.display();

}

int main() {

cout << "=Chocolate=" << endl;

Chocolate cherry("cherry");

Chocolate orange("orange");

cout << "\n=Box=" << endl;

Box b;

cout << "\n=++=" << endl;

b += orange;

b += cherry;

cout << "\n=show=" << endl;

show(b);

cout << "\n=done=" << endl;

return 0;

}

**1- Explain what the operator** Box& operator+=(const Chocolate& c) **does.**

**2- How can you improve the performance the** show**() function in this program. Explain the effect of your upgrade in one or two sentences.**

**3- Explain in one or two sentences the meaning of the keyword const in the** display() **functions?**

**4- Describe in one or two sentences the effect of removing the default parameter value in the definition of the** Chocolate() **constructor.**