**Part A: Concepts [ total of 15 marks]**

Using your own words, briefly answer the following questions:

1. Explain the difference between declaring the data members as private vs public. **[2 Marks]**
2. What is a Namespace? What does the line of code “using namespace std;” do? Where should this line of code not be included? **[4 Marks]**
3. True or False. If you do not declare a default constructor, the compiler will assign garbage values to your class data members. **[1 Marks]**
4. Explain what memory leak is and how to prevent it.**[2 Marks]**
5. When do you have to implement an operator function as a ***helper?*** **[2 Marks]**
6. Write the piece of code/s you need to prohibit copying and/or copy assigning of a class named Book? **[4 Marks]**

**Part B: Debugging [Total of 15 Marks]**

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;

3.

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:

**Part C: Programming [Total of 30 Marks]**

**struct Room {**

**char m\_name[5];**

**double m\_volume;**

**};**

Using the structure “Room” above, complete the following class named Department . Your class represents a set of Rooms in a department and your design meets the following specifications. The class has

two attributes:

* + m\_room — a pointer to a dynamically allocated array of Rooms.
  + m\_size — the number of rooms in the Department.
* a default constructor that sets the Department to a safe empty state
* a constructor with one parameter for number of rooms. This constructor dynamically allocates an array of rooms to the number rooms received through the parameter and keeps its address in the m\_room pointer.
* An overload the += operator for a double as the right operand. This will increase the size of each room in the department by the amount of the double operand.
* An overloaded == operator that takes two Departments operands and compare them if they are identical. Two Departments are the same if they have the same rooms and two rooms are the same if they share the same name and volume.

- If invalid data is received all of the above will set the object to a safe empty state.

- For full marks:

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

- Use Helper functions when needed (depending on the functionality and prototype )

- Make sure that your implementation does not create memory leaks.

- Write the prototypes in the class below and the implementations after the class code.

class Department {

// data members go here

public:

//sets the class to the values recieved from the console

istream& set(istream& is=cin) {

if (m\_room) {

for (int i = 0; i < m\_size; i++) {

is >> m\_room[i];

}

}

return is;

}

// dispays the class on the console

ostream& display(ostream& os=cout)const {

if (m\_room) {

os << "Bld | No" << endl

<< "----|----" << endl;

for (int i = 0; i < m\_size; os<<m\_room[i++]);

os << "---------" << endl;

}

return os;

}

// constructors and member functions prototypes

};

// helper functions prototypes if needed

// constructors and functions implementation