National University of Computer and Emerging Sciences

Islamabad Campus

Object Oriented Programming (CS1004)

Sessional-II Exam

Date: November 8th 2025

Total Time (Hrs): 1

Total Marks: 60

Total Questions: 4

Course Instructor(s)

Ms. Momina Behzad

Model Solution

Roll No _____ Section _____ Student Signature _____

Do not write below this line

Attempt all the questions.

[CLO 2: Apply OOP concepts(Encapsulation, Inheritance, Polymorphism, Abstraction) to computing problems for the related program]

Q1: Trace the following game to find output. Suppose the code is error free. Attempt this question on answer sheet. **[15 marks]**

```
#include <iostream>
#include <string>
using namespace std;
class Shadow {
   int val;
   string tag;
   static int total;
public:
   Shadow(int v = 0, string t = "X") {
      val = v;
      tag = t;
      total++;
      cout << "Created " << tag << " (" << val << ") | Total: " << total << endl;
  }
  Shadow(const Shadow& s) {
     val = s.val;
     tag = s.tag + "_copy";
     total++;
     cout << "Copied " << s.tag << " -> " << tag << " | Total: " << total << endl;
}
~Shadow() {
   cout << "Destroyed " << tag << " | Remaining: " << --total << endl;
}
Shadow& operator=(const Shadow& s) {
   if (this != &s) {
      val = s.val;
```

```
    tag = s.tag + "_asgn";
    cout << "Assigned " << s.tag << " -> " << tag << endl;
  }
  return *this;
}
Shadow operator+(const Shadow& s) const {
  Shadow t(val + s.val, tag + "+" + s.tag);
  cout << "Added " << tag << " and " << s.tag << " -> " << t.tag << endl;
  return t;
}
Shadow operator++(int) {
  Shadow temp = *this;
  val++;
  tag += "_inc";
  cout << "Post++ " << temp.tag << " -> " << tag << " (" << val << ")" << endl;
  return temp;
}
bool operator>(const Shadow& s) const {
  cout << "Compare " << tag << "(" << val << ") > " << s.tag << "(" << s.val << ")";
  return val > s.val;
}
static void report() {
  cout << "Alive: " << total << endl;
}

friend void fusion(const Shadow&, const Shadow&);
};
int Shadow::total = 0;
void fusion(const Shadow& a, const Shadow& b) {
  cout << "Fusion of " << a.tag << " & " << b.tag << " = " << (a.val + b.val) << endl;
}
int main() {
  cout << "--- Phase 1 ---" << endl;
  Shadow s1(2, "Alpha"), s2(5, "Beta");
  Shadow s3 = s1 + s2;
  s3++;
  Shadow s4 = s3 + s1;
  Shadow::report();
  cout << "--- Phase 2 ---" << endl;
  cout << "-------For 5 bonus marks find the error in this line of code-------" << endl
  Shadow s5(s4);
  s1 = s5;
  if (s2 > s3)
    cout << "Beta stronger\n";
  else
    cout << "Alpha+Beta stronger\n";
  fusion(s2, s3);
    Shadow::report();
  cout << "--- Phase 3 (Temp Zone) ---" << endl;
```

∴ only for correct semicolon

; → 5 bonus marks

→ if not mentioned here ⇒ zero

→ if wrong ⇒ zero.

National University of Computer and Emerging Sciences

Islamabad Campus

```
Shadow x1(10, "TempA"), x2(3, "TempB");
Shadow x3 = x1 + x2;
  x3++;
  fusion(x1, x3);
  if (x1 > x2) {
     cout << "TempA leads\n"; }
  else{
     cout << "TempB wins\n"; }
  Shadow::report();
 cout << "--- Cleanup ---" << endl;
Shadow::report();
cout << "End of program.\n";
return 0;
}
```

→ Total lines of output = 30 so each line carry 0.5 marks if executed properly

→ If there is a single error in an output statement than zero marks for that particular statement.

[CLO 1: Demonstrate the basic concepts of OOP] **[20 marks]**

Q2: Classify the following statements as true or false.

1. A static member function can access non-static data members directly if it belongs to the same class. F
2. The friend keyword violates encapsulation completely because it makes every member of one class public to another. T
3. Operator overloading can create new operators that don't exist in C++. F
4. In operator overloading, at least one operand of an overloaded operator must be a user-defined type. T
5. Overloading the = operator is mandatory when defining copy constructors. F
6. Friend functions are called using the dot operator. F
7. Two different classes can share the same friend function. T
8. A static data member has only one copy per object of a class. T
9. Overloading << and >> operators requires making them friend functions in most cases. T
10. Composition implies that the lifetime of the contained object depends on the lifetime of the container. T
11. In aggregation, if the container object is destroyed, the contained objects are destroyed automatically. F
12. Association represents a "has-a" relationship that always involves ownership. F
13. A friend class can access private and protected members of another class. T
14. Operator overloading changes the operator's precedence in C++. F
15. A static member function can be called using the class name, even when no object exists. T
16. In composition, an object cannot exist independently of the class it is part of. T
17. A static data member is created for every instance of a class separately. F
18. The this pointer can be used inside static member functions. F
19. Association always implies bi-directional linkage between two classes. F
20. In aggregation, the contained objects can exist even if the container is destroyed. T

Q1

--- Phase 01 ---

Created Alpha (2) | Total: 01

Created Beta (5) | Total: 02

Created Alpha + Beta (7) | Total: 03

Added Alpha & beta → Alpha + Bela

Copied Alpha + Beta → Alpha + Beta_copy | Total: 04

Post ++ Alpha + Beta_copy → Alpha + Beta_inc (8)

Destroyed alpha + beta_copy → | Remaining 03

Created Alpha + beta_inc + Alpha (10) | Total: 04

Added Alpha + Beta_inc and Alpha → Alpha + Beta_inc + Alpha

Alive 04

--- Phase 02 ---

copied Alpha + Beta_inc + Alpha → Alpha + Beta_inc + Alpha_copy

Total: 05

Assigned Alpha + Beta_inc + Alpha_copy → Alpha + Beta_inc + Alpha_copy_assign.

compare Beta (5) > Alpha + Beta_inc (8) Alpha + Beta stronger

Fusion of Beta & Alpha + Beta_inc = 13

Alive: 05

--- Phase 03 (TempZone) ---

For 5 bonus marks find error in this line of code.

Created TempA (10) | Total: 06

Created TempB (3) | Total: 07

Created tempA + tempB (13) | Total: 08

Added TempA & TempB → TempA + TempB

Copied TempA + TempB → TempA + TempB_copy | Total : 9

Post ++ TempA + TempB_copy → TempA + TempB_inc (14)

Destroyed TempA + TempB_copy | Remaing : 08

Fusion of TempA & TempA + TempB_inc = 24

Compare TempA (10) > TempB (3) TempA leads.

Alive 08

— — — — Cleanup — — — — — —

Alive : 08

End of program

Destroyed TempA + TempB_inc | Remaing : 07

Destroyed TempB | Remaing 06

Destroyed TempA | Remaing : 05

Destroyed Alpha + Beta_inc + Alpha_copy | Remaing : 04

Destroyed Alpha + Beta_inc + Alpha | Remaing : 03

(3)

Destroyed Alpha + Beta_inc | Remaing

or

Destroyed Beta | Remaing or

Destroyed Alpha + Beta_inc

+ Alpha_copy_asg

| Remaing (3)

[CLO 2: Apply OOP concepts(Encapsulation, Inheritance, Polymorphism, Abstraction) to computing problems for the related program]

Q3: Carefully examine each of the following code snippets. **Determine** whether the code is syntactically and logically correct. If any snippet contains an error, rewrite the corrected version. If the snippet is error-free, clearly state that "This code does not contain any errors." **[2.5*4=10 marks]**

Code	Corrected Code

Part A:
class Demo {
  int data;
public:
  static void show() {
  cout << data;
  }
};
class Demo
{
int data;
public
void show()
cout<<data;
}
	The static function can not access non static data member.

Part B:
class Ghost {
  int val;
  friend void reveal(Ghost g);
public:
  Ghost(int v):val(v){}
};
int main() {
  Ghost g(10);
  reveal(g);
}
friend void reveal (Ghost &g);
	should be passed through reference.

Part C:
class Point {
public:
  int x;
  Point(int a=0):x(a){}
  int operator+(Point &p) { return x + p.x; }
};
int main() {
  Point p1(2), p2(3);
  cout << p1.operator+(p2) << endl;
  cout << p1.operator+(5) << endl;
}
add
	int operator+(int a) {return x+a;}
	No operator for this.

Part D:
class Box {
  int x;
public:
  Box(int a=0):x(a){}
  operator+(Box b) {
  return x + b.x;
  }
};
int main() {
  Box b1(3), b2(4);
  Box b3 = b1 + b2;
  cout << "Done";
}
return type is missing.
Box operator+(Box b)
return x+b.x;	

[CLO 3: Model an algorithmic solution for a given problem using OOP.]

Q4: Each Teacher can teach multiple Courses, and each course can be taught by only one teacher at a time. Teachers can also guide students for final-year projects. Each Student can enroll in multiple courses. Each student also has a Profile containing personal details such as address, date of birth, and contact information. If the student account is deleted, their profile should also be removed from the system. Each Course has a set of Assignments, and those assignments belong *only* to that course. Once the course is deleted, all its assignments are removed automatically. Each Classroom contains various Equipment (like projector, whiteboard, microphone). The same equipment can be shifted between classrooms if needed. The Administration module monitors all courses and classrooms and can assign teachers to courses and allocate classrooms for lectures. **Demonstrate** the following tasks on above

1. Identify the classes and their relationships. [05]
2. Create suitable classes with meaningful attributes and methods. [05]
3. Implement operator overloading in at least one class. [03]
4. Write short test code (main) demonstrating how objects interact in this system. [02]

[15 marks]

Teacher ⟶ Course (Association)
Student ⟶ Course (Association)
Student ⟶ profile (composition)
Course ⟶ Assignment (composition)
Classroom ⟶ Equipment (Aggregation)

→ Each Relationship carry 01 mark.
→ In part 02 if a class is properly created (while maintaing its relationships than 01 mark otherwise zero)
→ In part 03 if one operator is properly overloaded then 03 otherwise zero.
→ In part 04 (if main shows proper use of relationships & overloading than 02 otherwise zero)