**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. Can a class inherit multiple classes in C++?
   1. Yes, but only up to three base classes
   2. Yes, but only if the base classes are in different namespaces
   3. No, C++ only supports single inheritance
   4. **Yes, C++ supports multiple inheritance**
2. What is the scope of a destructor in C++?
   1. Local to the class
   2. Global in the program
   3. **Limited to the object's lifetime**
   4. Limited to the class's lifetime
3. Can constructors be virtual in C++?
   1. Yes
   2. **No**
   3. Only if they are private
   4. Only if they are static
4. What is the purpose of a pure virtual function in C++?
   1. To prevent objects from being created from a class
   2. **To force derived classes to override the function**
   3. To deallocate memory before the object is destroyed
   4. To define the default behavior of a class member
5. What is the difference between an abstract class and an interface in C++?
   1. An abstract class cannot have constructors, while an interface can.
   2. An abstract class can have data members, while an interface cannot.
   3. **An abstract class can have both concrete and pure virtual functions, while an interface can only have pure virtual functions.**
   4. An abstract class can be inherited, while an interface cannot.
6. How is a destructor identified in C++?
   1. **By the '~' symbol followed by the class name**
   2. By the '+' symbol followed by the class name
   3. By the '^' symbol followed by the class name
   4. By the '-' symbol followed by the class name
7. When should you use a virtual destructor in a base class?
   1. When the base class has no derived classes
   2. **When the base class has at least one virtual function**
   3. When the base class is used to create objects
   4. When the base class is a singleton
8. What is the purpose of the 'this' pointer in C++?
   1. To access static class members
   2. **To access the object's address inside a member function**
   3. To call the base class constructor
   4. To deallocate memory for an object
9. Can a destructor be overloaded in C++?
   1. Yes
   2. **No**
   3. Only if the class has multiple constructors
   4. Only if the class has multiple member functions
10. What happens if a class has both a copy constructor and a copy assignment operator?
    1. The program will not compile.
    2. **The copy constructor will be called during assignment.**
    3. The copy assignment operator will be called during object creation.
    4. The compiler will randomly choose which one to call.

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

1. Which operator should be overloaded in the following code to make the program error free?

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 == b2){

cout<<"Equal";

}

else{

cout<<"Not Equal";

}

return 0;

}

a) +

**b) ==**

c) =

d) ()

1. What is the output of the following code?

#include <iostream>

using namespace std;

class A {

public:

A() { cout << "A constructor" << endl; }

~A() { cout << "A destructor" << endl; }

};

class B : public A {

public:

B() { cout << "B constructor" << endl; }

~B() { cout << "B destructor" << endl; }

};

class C : public B {

public:

C() { cout << "C constructor" << endl; }

~C() { cout << "C destructor" << endl; }

};

int main() {

C c;

return 0;

}

**a) A constructor, B constructor, C constructor, C destructor, B destructor, A destructor**

b) C constructor, B constructor, A constructor, A destructor, B destructor, C destructor

c) C constructor, B constructor, A constructor, A destructor, B destructor

d) A constructor, B constructor, C constructor, A destructor, B destructor, C destructor

1. When base class is derived in protected mode, then\_\_\_\_\_\_\_\_\_\_\_\_\_ .

1. public members of base class become private members of derived class.

2. public members of base class become protected members of derived class.

3. public members of base class become public members of derived class.

4. protected members of base class become protected members of derived class.

5. protected members of base class become private members of derived class.

6. protected members of base class become public members of derived class.

1. Only 1, 5
2. Only 1, 6
3. Only 2, 6
4. **Only 2, 4**
5. Give the function prototype of the operator function which we need to define in this program so that the program has no errors.

#include <iostream>

#include <string>

using namespace std;

class Box{

int capacity;

public:

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

int main(int argc, char const \*argv[])

{

Box b1(10);

Box b2 = Box(14);

if(b1 == b2){

cout<<"Equal";

}

else{

cout<<"Not Equal";

}

return 0;

}

1. bool operator==();
2. bool operator==(Box b){}
3. **bool operator==(Box b);**
4. Box operator==();
5. In case of inheritance where both base and derived class are having constructor and destructor, then which if the following are true ?

1. Constructors are executed in their order of derivation

2. Constructors are executed in reverse order of derivation

3. Destructors are executed in their order of derivation

4. Destructors are executed in reverse order of derivation

1. Only 2 ,4
2. Only 1 , 3
3. **Only 1 , 4**
4. Only 2, 3

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) You have been tasked with designing a C++ class called "Box" that represents a three-dimensional box.

Your goal is to create a class with multiple constructors to initialize its dimensions and provide a method to display the volume of the box.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | Box box1 | box2(3, 4, 5) | Box box3(2) |
| **Output** | Volume of box1: 0 | Volume of box2: 60 | Volume of box3: 8 |

Solution :

**#include <iostream>**

**class Box {**

**private:**

**int length;**

**int width;**

**int height;**

**public:**

**// Default Constructor**

**Box() {**

**length = 0;**

**width = 0;**

**height = 0;**

**}**

**// Parameterized Constructor**

**Box(int l, int w, int h) {**

**length = l;**

**width = w;**

**height = h;**

**}**

**// Constructor with a single argument**

**Box(int side) {**

**length = width = height = side;**

**}**

**int calculateVolume() {**

**return length \* width \* height;**

**}**

**};**

**int main() {**

**Box box1; // Default constructor**

**Box box2(3, 4, 5); // Parameterized constructor**

**Box box3(2); // Constructor with a single argument**

**std::cout << "Volume of box1: " << box1.calculateVolume() << std::endl;**

**std::cout << "Volume of box2: " << box2.calculateVolume() << std::endl;**

**std::cout << "Volume of box3: " << box3.calculateVolume() << std::endl;**

**return 0;**

**}**

Q17) Imagine you are a software developer working on a project called "The Romantic Date Planner."

The application aims to assist couples in planning and organizing memorable dates. As part of the project, you need to create a class that represents a date, allowing users to compare different dates easily.

Your task is to develop a C++ class called "Date" that will be the cornerstone of the Romantic Date Planner application.

The "Date" class should have the ability to compare two dates using the '==' and '!=' operators.

The application will use these operators to determine whether two dates are the same or different, respectively.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | d1(15, 7, 2023)  d2(15, 7, 2023) | d1(15, 7, 2023)  d2(5, 4, 2021) | d1(27, 11, 2003)  d2(27,11, 2003) |
| **Output** | d1 and d2 are the same dates. | d1 and d2 are different dates. | d1 and d2 are the same dates. |

Solution :

**#include <iostream>**

**using namespace std;**

**class Date {**

**private:**

**int day;**

**int month;**

**int year;**

**public:**

**Date(int d, int m, int y) : day(d), month(m), year(y) {}**

**// Overloading the equality '==' operator**

**bool operator==(const Date& other) {**

**return day == other.day && month == other.month && year == other.year;**

**}**

**// Overloading the inequality '!=' operator**

**bool operator!=(const Date& other) {**

**return !(\*this == other);**

**}**

**void display() {**

**cout << day << "/" << month << "/" << year << endl;**

**}**

**};**

**int main() {**

**Date d1(15, 7, 2023);**

**Date d2(15, 7, 2023);**

**Date d3(1, 1, 2023);**

**if (d1 == d2) {**

**cout << "d1 and d2 are the same dates." << endl;**

**} else {**

**cout << "d1 and d2 are different dates." << endl;**

**}**

**if (d1 != d3) {**

**cout << "d1 and d3 are different dates." << endl;**

**} else {**

**cout << "d1 and d3 are the same dates." << endl;**

**}**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) You have been tasked with creating a grocery list manager program for a local supermarket.

The program should allow users to create and manipulate their grocery lists efficiently.

To achieve this, you need to create a class called GroceryList representing an array, and overload the '[]' operator

to enable easy access to array elements.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | sugar rice oil bread salt | potato masala oil pasta wheat | carrot flour mayo rice corn |
| **Output** | sugar rice oil bread salt | potato masala oil pasta wheat | carrot flour mayo rice corn |

Solution :

**#include <iostream>**

**#include <stdlib.h> // standard library function**

**using namespace std;**

**const int SIZE = 5;**

**class MyArray {**

**private:**

**string arr[SIZE];**

**public:**

**MyArray() {**

**for (int i = 0; i < SIZE; i++) {**

**cin>>arr[i];**

**}**

**}**

**// Overloading the array index '[]' operator**

**string& operator[](int index) {**

**if (index < 0 || index >= SIZE) {**

**cout << "Index out of range." << endl;**

**exit(1);**

**}**

**return arr[index];**

**}**

**void display() {**

**for (int i = 0; i < SIZE; i++) {**

**cout << arr[i] << " ";**

**}**

**cout << endl;**

**}**

**};**

**int main() {**

**MyArray arr;**

**arr.display();**

**return 0;**

**}**