**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. Which of the following is not a type of Constructor?
   1. **Friend constructor**
   2. Copy constructor
   3. Default constructor
   4. Parameterized constructor
2. Which of the following is correct?
   1. Base class pointer object cannot point to a derived class object
   2. **Derived class pointer object cannot point to a base class object**
   3. A derived class cannot have pointer objects
   4. A base class cannot have pointer objects
3. In case of non-static member functions how many maximum object arguments a binary operator overloaded function can take?
   1. **1**
   2. 2
   3. 3
   4. 0
4. Which of the following class allows to declare only one object of it?
   1. Abstract class
   2. Virtual class
   3. **Singleton class**
   4. Friend class
5. Out of the following, which is not a member of the class?
   1. Static function
   2. **Friend function**
   3. Constant function
   4. Virtual function
6. Which of the following statements about friend classes in C++ is correct?
   1. **Friend classes have access to the private members of the class they are friends with**
   2. Friend classes are inherited along with the base class
   3. Friend classes can be used to achieve multiple inheritance
   4. Friend classes must be defined within the scope of the base class
7. Which constructor is called when an object is created using the default constructor syntax?
   1. Copy constructor
   2. Parameterized constructor
   3. Destructor
   4. **Default constructor**
8. Which of the following is a static polymorphism mechanism?
   1. Function overloading
   2. Operator overloading
   3. Templates
   4. **All of the mentioned**
9. Can a constructor be virtual in C++?
   1. Yes
   2. **No**
10. What is the purpose of the "virtual" keyword in C++?
    1. It is used to make a member variable of a class constant
    2. **It is used to allow a function in the base class to be overridden in the derived class**
    3. It is used to create a pointer to an object
    4. It is used to prevent inheritance from a class

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

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

1. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

void print(int i)

{

cout << i;

}

void print(double f)

{

cout << f;

}

int main(void)

{

print(5);

print(500.263);

return 0;

}

* 1. **5500.263**
  2. 500.2635
  3. 500.263
  4. 500.266

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A{

mutable int a;

public:

A(){

cout<<"Default constructor called\n";

}

A(const A& a){

cout<<"Copy Constructor called\n";

}

};

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

{

A obj;

A a1 = obj;

A a2(obj);

}

1. Default constructor called

Copy Constructor called

1. **Default constructor called**

**Copy Constructor called**

**Copy Constructor called**

1. Default constructor called

Default constructor called

Copy Constructor called

1. Copy Constructor called

Default constructor called

Copy Constructor called

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A{

mutable int a;

public:

A(){

cout<<"A's Constructor called\n";

}

~A(){

cout<<"A's Destructor called\n";

}

};

class B: public A{

public:

B(){

cout<<"B's Constructor called\n";

}

~B(){

cout<<"B's Destructor called\n";

}

};

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

{

B b1;

}

a)

A's Constructor called

B's Constructor called

b)

A's Destructor called

B's Destructor called

**c)**

**A's Constructor called**

**B's Constructor called**

**B's Destructor called**

**A's Destructor called**

d)

A's Constructor called

B's Constructor called

A's Destructor called

B's Destructor called

1. What will be the output of the following C++ code?

#include <iostream>

#include <string>

using namespace std;

class A

{

mutable int a;

public:

int assign(int i) const {

a = i;

}

int return\_value() const {

return a;

}

};

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

{

A obj;

obj.assign(5);

cout<<obj.return\_value();

}

1. **5**
2. Error
3. Segmentation fault
4. Undefined value
5. What happens when objects s1 and s2 are added?

string s1 = "Hello";

string s2 = "World";

string s3 = (s1+s2).substr(5);

1. Error because s1+s2 will result into string and no string has substr() function
2. Segmentation fault as two string cannot be added in C++
3. **The statements runs perfectly**
4. Run-time error

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

Q16) Imagine you are a mathematician working on a geometry project. You need to calculate the volume of various cubes for your research. To simplify the process, you want to create a C++ class to represent a cube and calculate its volume. Give default size as 1 to cube.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | side=blank value | side= 3 | side=5 |
| **Output** | Volume of cube: 1 | Volume of cube: 27 | Volume of cube2: 125 |

Solution :

**#include <iostream>**

**class Cube {**

**private:**

**int side;**

**public:**

**// Constructor with default argument**

**Cube(int s = 1) {**

**side = s;**

**}**

**int calculateVolume() {**

**return side \* side \* side;**

**}**

**};**

**int main() {**

**Cube cube1; // Uses default argument, side = 1**

**Cube cube2(5);**

**std::cout << "Volume of cube1: " << cube1.calculateVolume() << std::endl;**

**std::cout << "Volume of cube2: " << cube2.calculateVolume() << std::endl;**

**return 0;**

**}**

Q17) Imagine you are a teacher at a school, and you need to calculate the average grade of a student based on their exam scores.

You want to create a C++ program to handle this task. Each student has multiple subjects, and the average grade will be calculated

as the sum of the grades divided by the total number of subjects.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | {80, 90, 85, 95, 70} | {85, 87, 76, 90, 60} | {81, 92, 75, 85, 79} |
| **Output** | Average grade of student: 84 | Average grade of student: 79.6 | Average grade of student: 82.4 |

Solution :

**#include <iostream>**

**#include <string>**

**class Student {**

**private:**

**std::string name;**

**int grades[5];**

**int numGrades;**

**public:**

**Student(const std::string& n, const int\* gradesArray, int num) : name(n), numGrades(num) {**

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

**grades[i] = gradesArray[i];**

**}**

**}**

**friend double calculateAverageGrade(const Student& stu);**

**};**

**double calculateAverageGrade(const Student& stu) {**

**if (stu.numGrades == 0) {**

**return 0.0;**

**}**

**int sum = 0;**

**for (int i = 0; i < stu.numGrades; ++i) {**

**sum += stu.grades[i];**

**}**

**return static\_cast<double>(sum) / stu.numGrades;**

**}**

**int main() {**

**const int gradesArray[] = {80, 90, 85, 95, 70};**

**const Student stu1("Alice", gradesArray, 5);**

**double averageGrade = calculateAverageGrade(stu1);**

**std::cout << "Average grade of student: " << averageGrade << std::endl;**

**return 0;**

**}**

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

Q18) You are working on a student record management system for a school, and you need to create a C++ program to compare the age of two students and find the older student based on their ages.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | student1=("Alice", 21)  student2=("Bob", 23) | student1=("Reha", 22)  student2=("Gaurav", 26) | student1=("Emma", 20)  student2=("Shiva", 19) |
| **Output** | Older Student: Bob, Age: 23 | Older Student: Gaurav, Age: 26 | Older Student: Emma, Age: 20 |

Solution :

**#include <iostream>**

**#include <string>**

**class Student {**

**public:**

**std::string name;**

**int age;**

**Student(const std::string& n, int a) : name(n), age(a) {}**

**};**

**// Function to find the older student and return a new Student**

**Student findOlderStudent(const Student& s1, const Student& s2) {**

**if (s1.age >= s2.age) {**

**return Student(s1.name, s1.age);**

**} else {**

**return Student(s2.name, s2.age);**

**}**

**}**

**int main() {**

**Student student1("Alice", 21);**

**Student student2("Bob", 23);**

**Student olderStudent = findOlderStudent(student1, student2);**

**std::cout << "Older Student: " << olderStudent.name << ", Age: " << olderStudent.age << std::endl;**

**return 0;**

**}**