**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. What is the access specifier used for class members to make them accessible within the class and its derived classes?
   1. public
   2. private
   3. **protected**
   4. static
2. In C++, can a derived class inherit multiple base classes?
   1. Yes, using single inheritance
   2. **Yes, using multiple inheritance**
   3. No, inheritance is limited to one base class
   4. Yes, using multilevel inheritance
3. Which of the following operator cannot be used to overload when that function is declared as friend function?
   1. -=
   2. ||
   3. ==
   4. **[]**
4. Which function is automatically called when an object is created?
   1. delete()
   2. destructor
   3. main()
   4. **constructor**
5. How do you define a constructor with no parameters in C++?
   1. **Constructor()**
   2. constructor(void)
   3. Constructor(void)
   4. Constructor::Constructor()
6. When is the destructor of a class called?
   1. When the object is created
   2. **When the object goes out of scope**
   3. When the object is passed to a function
   4. When the destructor is explicitly called
7. Can constructors be virtual in C++?
   1. Yes
   2. **No**
8. What is the syntax of overloading operator + for class A?
   1. **A operator+(argument\_list){}**
   2. A operator[+](argument\_list){}
   3. int +(argument\_list){}
   4. int [+](argument\_list){}
9. What is the return type of the copy constructor?
   1. void
   2. int
   3. **The same class object (by reference)**
   4. The same class object (by value)
10. Which of the following cannot be a friend?
    1. Function
    2. Class
    3. **Object**
    4. Operator function

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

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

1. Which of the following feature of OOPs is not used in the following C++ code?

class A{

int i;

public:

void print() {cout << "hello" << i;}

}

class B : public A{

int j;

public:

void assign (int a ) {k = a;}

}

a) Abstraction

b) Encapsulation

c) Inheritance

**d) Polymorphism**

1. What is the output of the following code?

#include <iostream>

using namespace std;

class Base {

public:

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

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

};

class Derived : public Base {

public:

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

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

};

int main() {

Base\* ptr = new Derived();

delete ptr;

return 0;

}

**a) Base constructor, Derived constructor, Derived destructor, Base destructor**

b) Derived constructor, Base constructor, Base destructor, Derived destructor

c) Base constructor, Derived constructor, Base destructor

d) Derived constructor, Base destructor

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

#include <iostream>

#include <string>

using namespace std;

class complex

{

int i;

int j;

public:

complex(){}

complex(int a, int b)

{

i = a;

j = b;

}

complex operator+(complex c)

{

complex temp;

temp.i = this->i + c.i;

temp.j = this->j + c.j;

return temp;

}

void show(){

cout<<"Complex Number: "<<i<<" + i"<<j<<endl;

}

};

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

{

complex c1(1,2);

complex c2(3,4);

complex c3 = c1 + c2;

c3.show();

return 0;

}

**a) Complex Number: 4 + i6**

b) Complex Number: 2 + i2

c) Error

d) Segmentation fault

1. What is the output of the following code?

#include <iostream>

using namespace std;

class MyClass {

public:

int x;

MyClass() : x(10) {}

};

int main() {

MyClass obj1;

MyClass obj2 = obj1;

cout << obj2.x << endl;

return 0;

}

**a) 10**

b) 0

c) Garbage value

d) Compiler error

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

#include <iostream>

#include <string>

using namespace std;

class Box

{

int capacity;

Box(){}

Box(double capacity){

this->capacity = capacity;

}

};

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

{

Box b1(10);

Box b2 = Box(14);

return 0;

}

**a) Error**

b) Segmentation fault

c) 4

d) No output

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

Q16) Create a class 'Rectangle' with attributes 'x', 'y' (coordinates of the top-left corner),'width', and 'height'.

Implement a function to check if two rectangles intersect.

Use objects to demonstrate rectangle intersection for two rectangles.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | rect1(0, 0, 4, 3)  rect2(2, 2, 5, 3) | rect1(0, 0, 4, 3)  rect2(5, 5, 9, 4) | rect1(1, 4, 6, 4)  rect2(3, 5, 6, 4) |
| **Output** | Rectangles intersect. | Rectangles do not intersect. | Rectangles intersect. |

Solution :

**#include <iostream>**

**class Rectangle {**

**private:**

**int x;**

**int y;**

**int width;**

**int height;**

**public:**

**// Constructor**

**Rectangle(int x\_, int y\_, int w, int h) : x(x\_), y(y\_), width(w), height(h) {}**

**// Function to check if two rectangles intersect**

**bool intersect(const Rectangle& other) {**

**if (x > other.x + other.width || other.x > x + width)**

**return false;**

**if (y > other.y + other.height || other.y > y + height)**

**return false;**

**return true;**

**}**

**};**

**int main() {**

**Rectangle rect1(1, 4, 6, 4);**

**Rectangle rect2(3, 5, 6, 4);**

**if (rect1.intersect(rect2))**

**std::cout << "Rectangles intersect.\n";**

**else**

**std::cout << "Rectangles do not intersect.\n";**

**return 0;**

**}**

Q17) You are a software developer working on an educational management system for a school.

The school administration wants to track the academic performance of their students efficiently.

To accomplish this, you decide to create a C++ class named Student that will store essential information about each student, including their name, roll number, and marks in five different subjects.

Implement methods to calculate the total and average marks of the student.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | {90, 85, 78, 92, 88} | {80, 89, 68, 90, 88} | {50, 69, 75, 48, 88} |
| **Output** | Total Marks: 433  Average Marks: 86.6 | Total Marks: 415  Average Marks: 83 | Total Marks: 330  Average Marks: 66 |

Solution :

**#include <iostream>**

**#include <string>**

**class Student {**

**private:**

**std::string name;**

**int rollNumber;**

**int marks[5];**

**public:**

**Student(const std::string& n, int roll, const int\* m) : name(n), rollNumber(roll) {**

**for (int i = 0; i < 5; ++i)**

**marks[i] = m[i];**

**}**

**int calculateTotal() {**

**int total = 0;**

**for (int i = 0; i < 5; ++i)**

**total += marks[i];**

**return total;**

**}**

**double calculateAverage() {**

**return static\_cast<double>(calculateTotal()) / 5;**

**}**

**};**

**int main() {**

**int marks[] = {50, 69, 75, 48, 88};**

**Student student("Alice", 1001, marks);**

**std::cout << "Total Marks: " << student.calculateTotal() << std::endl;**

**std::cout << "Average Marks: " << student.calculateAverage() << std::endl;**

**return 0;**

**}**

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

Q18) Create a base class NumberSeries with a method to generate the next term in the series.

Derive two classes FibonacciSeries and PrimeSeries from NumberSeries to generate the next Fibonacci number and

the next prime number, respectively.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | n=10 | n=5 | n=4 |
| **Output** | Fibonacci Series: 1 2 3 5 8 13 21 34 55 89  Prime Series: 3 4 5 7 11 13 17 19 23 29 | Fibonacci Series: 1 2 3 5 8  Prime Series: 3 4 5 7 11 | Fibonacci Series: 1 2 3 5  Prime Series: 3 4 5 7 |

Solution :

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX\_SIZE 100**

**int adjMatrix[MAX\_SIZE][MAX\_SIZE];**

**int visited[MAX\_SIZE];**

**struct queue**

**{**

**int size;**

**int f;**

**int r;**

**int\* arr;**

**};**

**//check if queue is empty**

**int isEmpty(struct queue \*q){**

**if(q->r==q->f){**

**return 1;**

**}**

**return 0;**

**}**

**//check if queue is full**

**int isFull(struct queue \*q){**

**if(q->r==q->size-1){**

**return 1;**

**}**

**return 0;**

**}**

**//add element in queue**

**void enqueue(struct queue \*q, int val){**

**if(isFull(q)){**

**printf("This Queue is full\n");**

**}**

**else{**

**q->r++;**

**q->arr[q->r] = val;**

**// printf("Enqued element: %d\n", val);**

**}**

**}**

**//remove element from queue**

**int dequeue(struct queue \*q){**

**int a = -1;**

**if(isEmpty(q)){**

**printf("This Queue is empty\n");**

**}**

**else{**

**q->f++;**

**a = q->arr[q->f];**

**}**

**return a;**

**}**

**void BFS(int start,int numVertices){**

**// Initializing Queue (Array Implementation)**

**struct queue q;**

**q.size = 400;**

**q.f = q.r = 0;**

**q.arr = (int\*) malloc(q.size\*sizeof(int));**

**// BFS Implementation**

**int node;**

**printf("%d ", start);**

**visited[start] = 1;**

**enqueue(&q, start); // Enqueue i for exploration**

**while (!isEmpty(&q))**

**{**

**int node = dequeue(&q);**

**for (int j = 0; j < numVertices; j++)**

**{**

**if(adjMatrix[node][j] ==1 && visited[j] == 0){**

**printf("%d ", j);**

**visited[j] = 1;**

**enqueue(&q, j);**

**}**

**}**

**}**

**}**

**int main(){**

**int numVertices;**

**printf("Enter the number of vertices: ");**

**scanf("%d", &numVertices);**

**printf("Enter the adjacency matrix:\n");**

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

**for (int j = 0; j < numVertices; j++) {**

**scanf("%d", &adjMatrix[i][j]);**

**}**

**}**

**// Initialize visited array**

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

**visited[i] = 0;**

**}**

**int startVertex;**

**printf("Enter the starting vertex: ");**

**scanf("%d", &startVertex);**

**BFS(startVertex, numVertices);**

**return 0;**

**}**