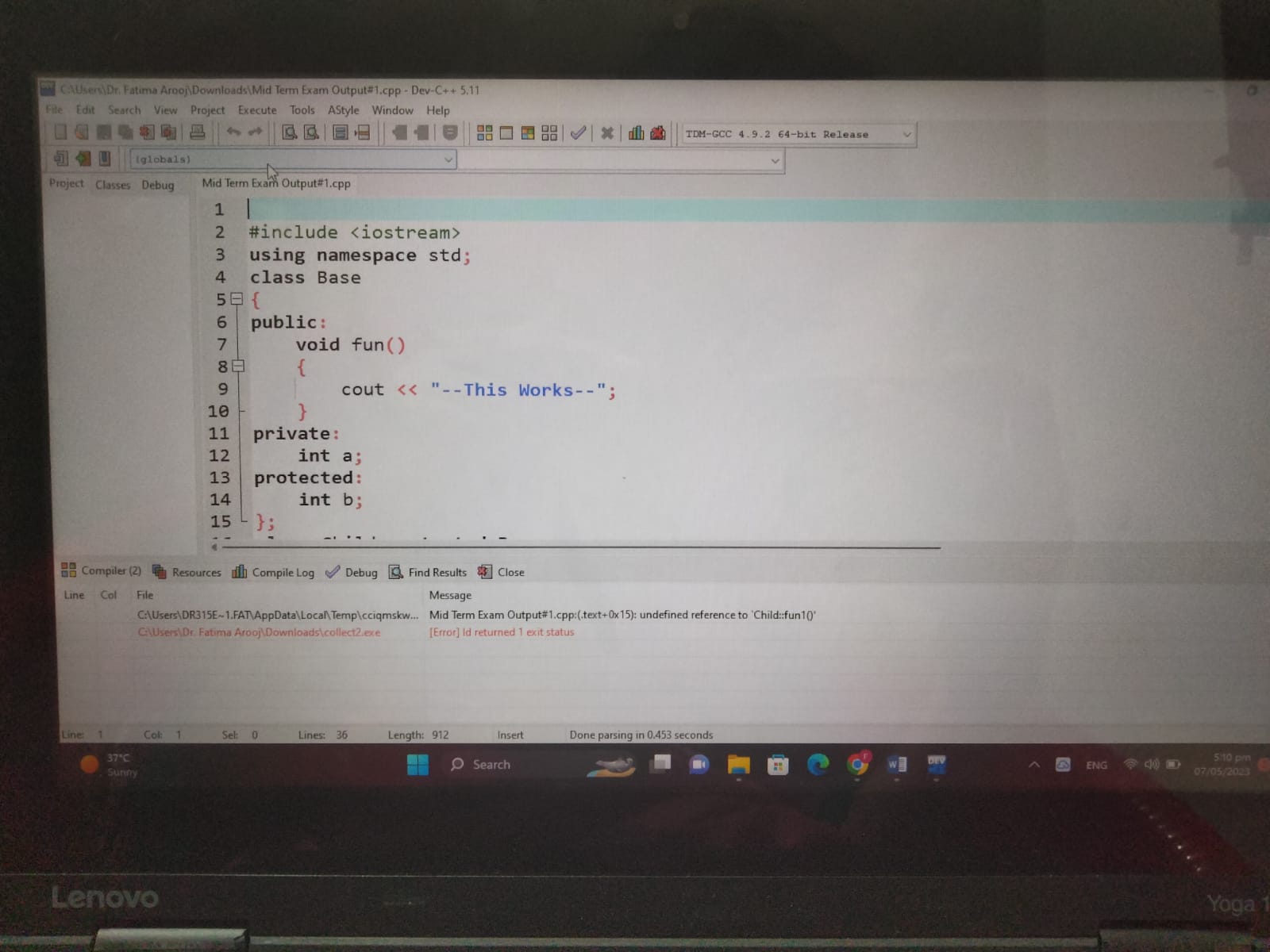
**QUESTION # 1:**

**PART A:**

**ANSWER;**

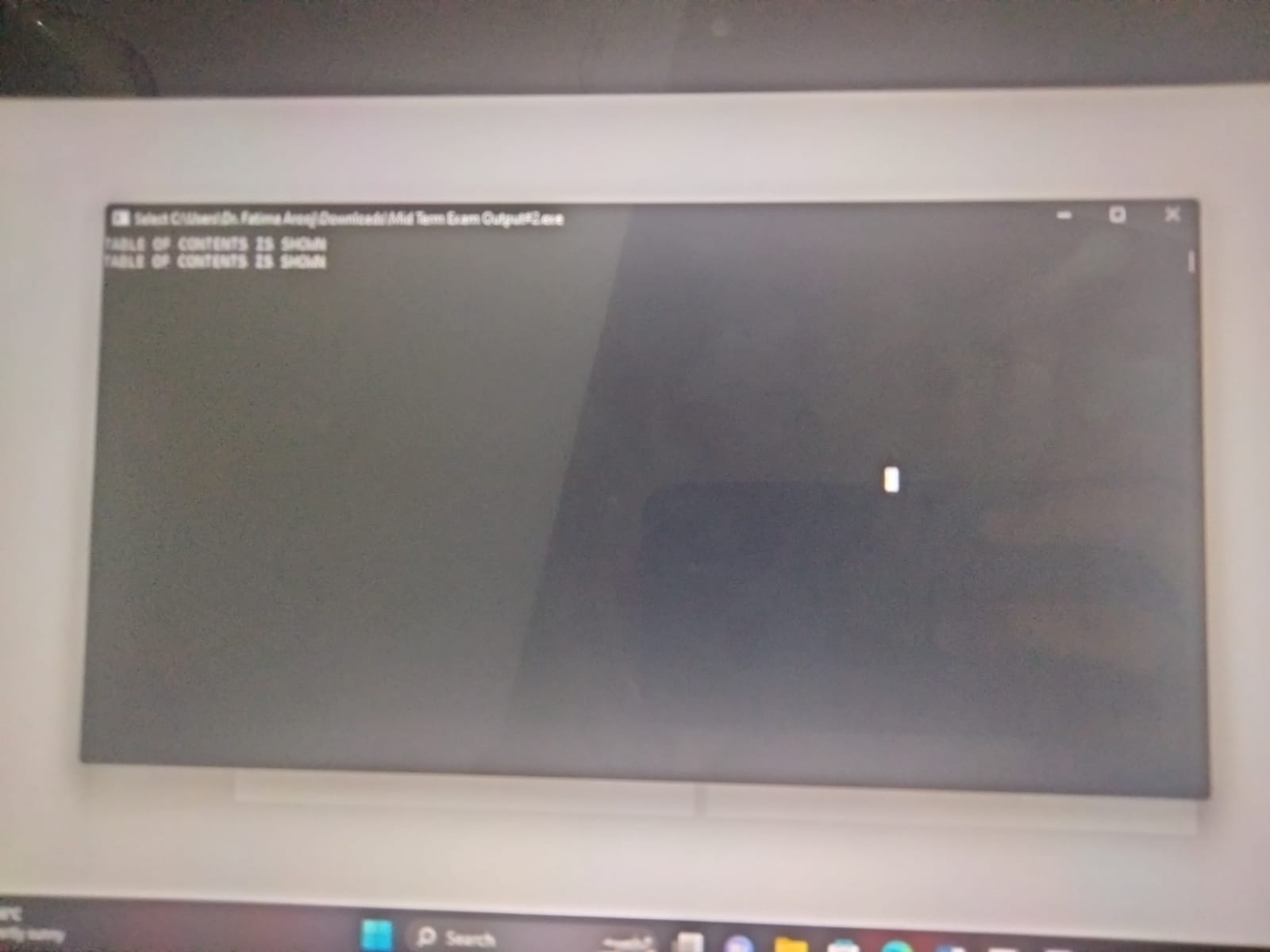
**ERROR**



**Part b:**

**ANSWER:**

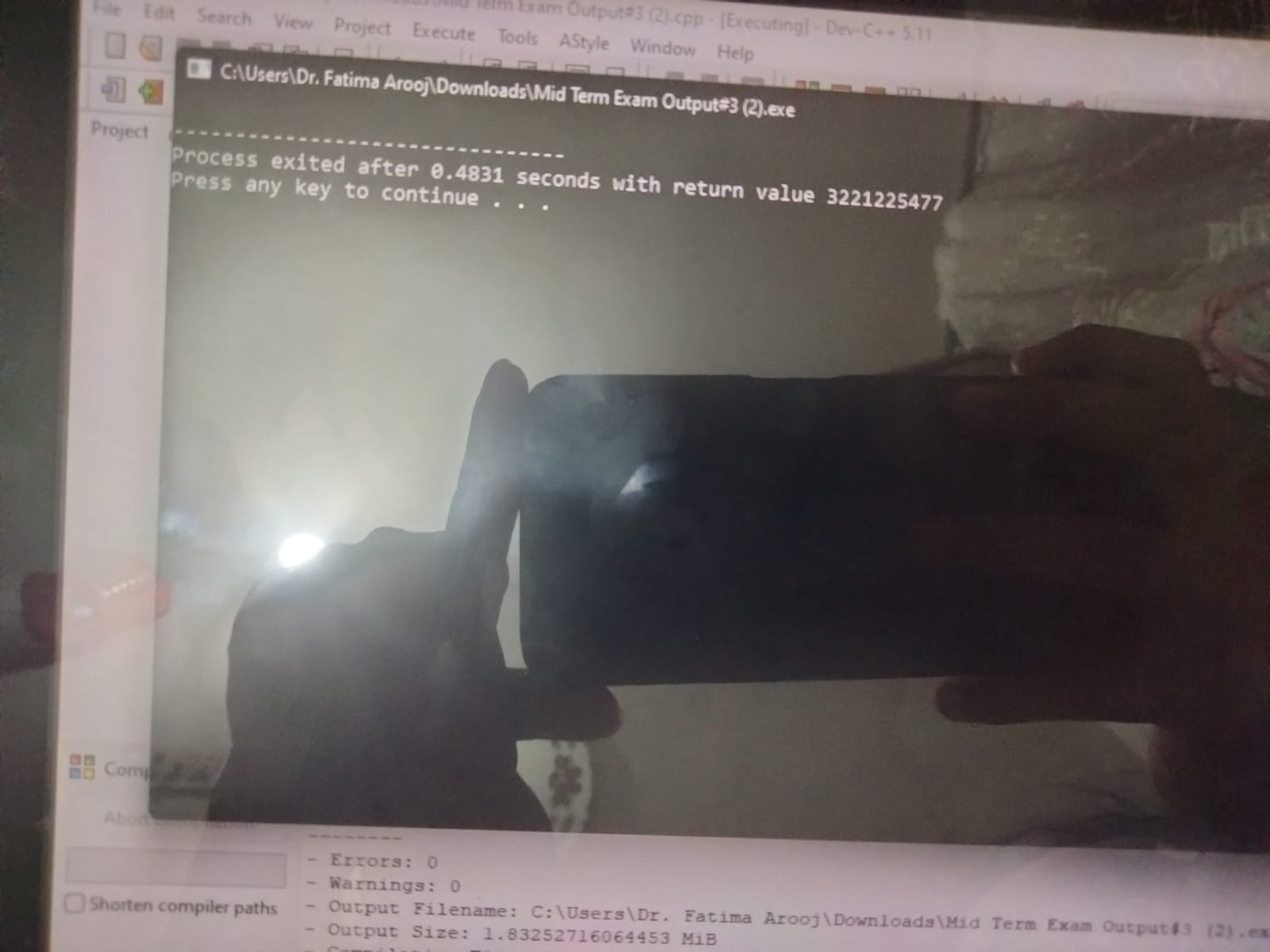
**TABLE OF THE CONTEN**

****

**PART C:**

**ANSWER:**

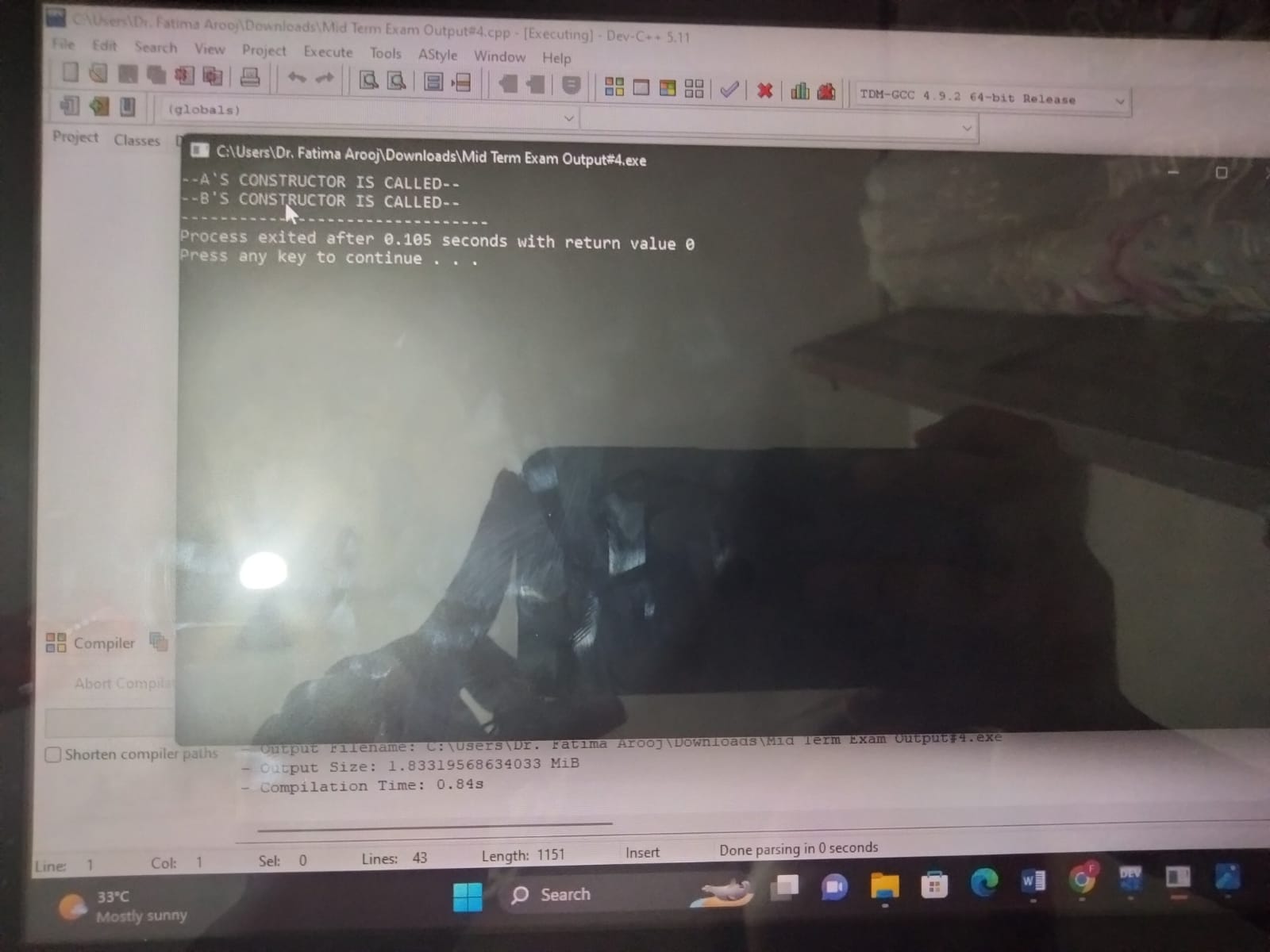
**0**

****

**PART D**

**ANSWER:**

**A’S CONSTRUCTOR IS CALLED**

****

**QUESTION NO 2:**

1. **What is the purpose of access modifiers in OOP languages?**

**PUBLIC:**The public access modifier is the direct opposite of the private access modifier. A class, method or variable can be declared as public and it means that it is accessible from any class. Public access modifier can be likened to a public school where anyone can seek admission and be admitted.

**PRIVATE:** Private access modifier is the most restrictive access level. Class and interfaces cannot be private. Variables that are declared private can be accessed outside the class, if public getter methods are present in the class.

**PROCTECTED:** It is a keyword. This access modifier is used to access the methods or data members of a class within the same package as well as outside the package but only through inheritance. The protected access modifier has more accessibility than private and defaults access modifiers.

**PART B)If we want to access the private members of a class in the child class what do we need to change?**

In object-oriented programming, private members of a class are not visible outside the class, including in child classes. However, there are certain cases where you might want to access private members of a class in a child class. In order to achieve this, you need to use the concept of inheritance and access modifiers.

To access private members of a parent class in a child class, you can change the access modifier of those members to protected. The protected access modifier allows the members to be accessed by the child class, as well as any other class that extends the parent class.

**PART C)**

a)NOT ACCESSIBLE

b)ACCESSIBLE

c)ACCESSIBLE

QUESTION 4:

**CODE :**

#include <iostream>

#include <string>

using namespace std;

class Character {

public:

int id;

string name;

int max\_power;

int strength;

Character(int id, string name, int max\_power, int strength) {

this->id = id;

this->name = name;

this->max\_power = max\_power;

this->strength = strength;

}

void walk() {

cout << name << " is walking." << endl;

}

void jump() {

cout << name << " is jumping." << endl;

}

void eat() {

cout << name << " is eating." << endl;

}

};

class Doremon : public Character {

public:

string gadgets[3];

string partner\_name;

Doremon(int id, string name, int max\_power, int strength, string gadgets[3], string partner\_name)

: Character(id, name, max\_power, strength) {

for (int i = 0; i < 3; i++) {

this->gadgets[i] = gadgets[i];

}

this->partner\_name = partner\_name;

}

void showGadgets() {

cout << "Gadgets: ";

for (int i = 0; i < 3; i++) {

cout << gadgets[i] << ", ";

}

cout << endl;

}

void launchAttack() {

cout << name << " is launching an attack!" << endl;

}

void fly() {

cout << name << " is flying." << endl;

}

};

class Benten : public Character {

public:

string watch\_name;

string powers[3];

int watch\_charge;

Benten(int id, string name, int max\_power, int strength, string watch\_name, string powers[3], int watch\_charge)

: Character(id, name, max\_power, strength) {

this->watch\_name = watch\_name;

for (int i = 0; i < 3; i++) {

this->powers[i] = powers[i];

}

this->watch\_charge = watch\_charge;

}

void rotateWatch() {

cout << "Rotating the " << watch\_name << " watch." << endl;

}

void fight() {

cout << name << " is fighting." << endl;

}

void drive() {

cout << name << " is driving." << endl;

}

};

int main() {

string gadgets[3] = {"Anywhere Door", "Takecopter", "Small Light"};

Doremon doremon(1, "Doremon", 100, 50, gadgets, "Nobita");

string powers[3] = {"Fireball", "Thunderbolt", "Big Bang"};

Benten benten(2, "Benten", 150, 75, "Omnitrix", powers, 50);

doremon.walk();

doremon.showGadgets();

doremon.launchAttack();

doremon.fly();

benten.jump();

benten.rotateWatch();

benten.fight();

benten.drive();

return 0;

}