Pre-Practical activity (Memo)

1) What is encapsulation?

This is how to restrict direct access to a component of an object through the use of access modifiers such private and public

2) What is inheritance and give an example?

When a child class/class uses the properties from another class known as the parent class.
An example would be a vehicle class with start() and stop() methods that would be accessible to a child class like car which gains access through inheritancei.

3) What is polymorphism and how does method overriding enable it?

Allows objects of different classes to be treated as objects of different classes to be treated as objects of a common parent class through method overriding

4) What is the difference between method overloading and method overriding? Provide an ex ample of each.

Method overriding occurs when a child class overrides a function from the parent class to provide a specific implementation for that class, and Method overloading occurs when multiple methods share the same name but have different parameter lists.

5) Why is it important to use access specifiers (like private, protected, public) in OOP? What would be the potential consequences of not using them properly?

Access specifiers are used to define the accessibility of properties in that class.

6) What is the output:

```
class Animal {
public:
    void makeSound() {
        cout << "Some generic animal sound" << endl;
    }
};

class Dog : public Animal {
public:
    void makeSound() {
        cout << "Bark!" << endl;
    }
};

int main() {
    Animal a;
    Dog d;
    a.makeSound();
    d.makeSound();
}
```

a) Some generic animal sound Bark!	b) Bark! Bark!
c) Some generic animal sound Some generic animal sound	d) Compile-time error

7)

```
class Vehicle {
protected:
    string make;
    string model;
public:
    Vehicle(string m, string mod) : make(m), model(mod) {}

    void showDetails() {
        cout << "Make: " << make << ", Model: " << model << endl;
    }
};

class Car : public Vehicle {
private:
    int numberOfDoors;
public:
    Car(string m, string mod, int doors) : Vehicle(m, mod),
numberOfDoors(doors) {}

    void showDetails() {
        cout << "Car - Make: " << make << ", Model: " << model << ", Doors:
" << numberOfDoors << endl;
    }
};

int main() {
    Car car("Toyota", "Corolla", 4);
    car.showDetails();
}
```

a) Make: Toyota, Model: Corolla	b) Car - Make: Toyota, Model: Corolla, Doors: 4
c) Car - Make: Toyota, Doors: 4	d) Compile-time error

8)

```
class Employee {
public:
    int salary;

    void setSalary(int s) {
        salary = s;
    }
};

int main() {
    Employee e;
    e.setSalary(5000);
    cout << e.salary << endl;
}
```

a) 5000	b) Undefined behaviour
c) 0	d) Compile-time error

7) In C++, create an animal class with the following attributes and functions
You must determine the appropriate data types

Animal	
Attributes	
name	the name of the animal
age	the age of the employee
Methods	
eat()	prints name + " is eating."
sound()	prints "Animal sound"

There are 2 subclasses/child classes of the Animal class Dog and Cat that both Override the Sound() function

Dog:

- Overrides Sound() and prints "woof!"

Cat:

- Overrides Sound() and prints "Meow! Meow!"

Create your Dog and Cat in the int main function which returns a 0
and perform the following:

- Set the name of the dog to "Jake" with the age 3 and the cat "Jane" with the age 2
- Make the Dog use the sound function
- Make the Dog use the eat method
- Make the cat use the sound function
- Make the cat use the eat method

Expected output:
Woof!
Jake is eating.
Meow! Meow!
Jane is eating.

```cpp
#include <iostream>
using namespace std;

class Animal {
protected:
    string name;
    int age;

public:
    Animal(string n, int a) : name(n), age(a) {}
    void eat() {
        cout << name << " is eating." << endl;
    }
    virtual void makeSound() {
        cout << "Animal sound." << endl;
    }
};
class Dog : public Animal {
public:
    Dog(string n, int a) : Animal(n, a) {}
    void makeSound() override {
        cout << "Woof!" << endl;
    }
};
class Cat : public Animal {
public:
    Cat(string n, int a) : Animal(n, a) {}
    void makeSound() override {
        cout << "Meow! Meow!" << endl;
    }
};

int main() {
    Dog dog("Jake", 3);
    Cat cat("Jane", 2);
    dog.makeSound();
    dog.eat();
    cat.makeSound();
    cat.eat();

    return 0;
}
```

8) In C++, create an Employee class with the following attributes and functions
You must determine the appropriate data types

Employee	
Attributes	
name	the name of the employee
grossSalary	the gross salary of the employee
Methods	
grossSalary	return grossSalary
netSalary	return netSalary

In the netSalary you are required to calculate the appropriate tax bracket based on the table below and deduct it from the grossSalary to get the netSalary value:

Taxable Income (R)	Tax Rate
0 - 237 100	18%
237 101 - 370 500	R42 678 + 26% of taxable income above 237 100
370 501 - 512 800	R77 362 + 31% of taxable income above 370 500
512 801 - 673 000	R121 475 + 36% of taxable income above 512 800
673 001 - 857 900	R179 147 + 39% of taxable income above 673 000
857 901 - 1 817 000	R251 258 + 41% of taxable income above 857 900
1 817 000 and above	R644 489 + 45% of taxable income above 1 817 000

Expected output
Employee Details:
Name: John Doe
Gross Salary: R400000
Net Salary: R314968

Manager Details:
Name: Alice
Department: IT
Gross Salary: R900000
Net Salary: R632323

```cpp
#include <iostream>
using namespace std;

class Employee {
protected:
    string name;
    int grossSalaryAmount;

public:
    Employee(string n, int grossSalary) : name(n),
grossSalaryAmount(grossSalary) {}

    int grossSalary() {
        return grossSalaryAmount;
    }
    int netSalary() {
        int taxableIncome = grossSalaryAmount;
        int tax = 0;

        if (taxableIncome <= 237100) {
            tax = taxableIncome * 0.18;
        } else if (taxableIncome <= 370500) {
            tax = 42678 + (taxableIncome - 237100) * 0.26;
        } else if (taxableIncome <= 512800) {
            tax = 77362 + (taxableIncome - 370500) * 0.31;
        } else if (taxableIncome <= 673000) {
            tax = 121475 + (taxableIncome - 512800) * 0.36;
        } else if (taxableIncome <= 857900) {
            tax = 179147 + (taxableIncome - 673000) * 0.39;
        } else if (taxableIncome <= 1817000) {
            tax = 251258 + (taxableIncome - 857900) * 0.41;
        } else {
            tax = 644489 + (taxableIncome - 1817000) * 0.45;
        }
        return grossSalaryAmount - tax;
    }
    virtual void showDetails() {
        cout << "Name: " << name << endl;
        cout << "Gross Salary: R" << grossSalary() << endl;
        cout << "Net Salary: R" << netSalary() << endl;
    }
};
```

```cpp
class Manager : public Employee {
private:
    string department;

public:
    Manager(string n, int grossSalary, string dept) : Employee(n,
grossSalary), department(dept) {}

    void showDetails() override {
        cout << "Name: " << name << endl;
        cout << "Department: " << department << endl;
        cout << "Gross Salary: R" << grossSalary() << endl;
        cout << "Net Salary: R" << netSalary() << endl;
    }
};

int main() {
    Employee emp("John Doe", 400000);
    Manager manager("Alice", 900000, "IT");

    cout << "Employee Details:" << endl;
    emp.showDetails();
    cout << endl;
    cout << "Manager Details:" << endl;
    manager.showDetails();

    return 0;
}
```