10/10

National University of Computer and Emerging Sciences (Lahore)			
Course:	OOP	Course code:	CS217
Section:	BSCS-2B	Semester:	Spring 2024
Duration:	40 minutes	TotalMarks:	10
Date:	6/May/2024	ID:	A
Name:	Zainab Saad	Roll no:	23L-0646

Question 1:

NOTE: Read the entire question first before attempting.

A construction company requires a paycheque management system. It employs different types of employees who are all paid differently as described below. All types of employees must have a function that calculates their salaries but an employee can only be one of three types; manager, engineer or salesperson. All employees "must" belong to at least one of these categories. An unclassified employee "cannot be paid a salary".

1. All `Employees` have the following attributes common:
   - `name` (string): The name of the employee.
   - `id` (int): The unique ID of the employee.
   - `baseSalary` (double):
2. Implement three derived classes: `Manager`, `Engineer`, and `Salesperson`, each inheriting from the `Employee` class with unique attributes:
   - For **`Manager`**: `department` (string), `bonus` (double), `calculateSalary()` method. They are paid a bonus in addition to their base salary.
   - For **`Engineer`**: `rate` (double), `numProjects` (int), `calculateSalary()` method. They are paid the product of their rate and no. of projects in addition to their base salary.
   - For **`Salesperson`**: `salesAchieved` (double), `commissionRate` (double), `calculateSalary()` method. They are paid a commission on each sale made in addition to base salary.
3. Implement default and parameterised constructors, destructors and a `calculateSalary()` method in each derived class to calculate the salary of the respective employee type based on the provided attributes.
4. Give output of the main given on the next page.

```
int main() {

    // Create employee objects
    vector<Employee*> employees;

    employees.push_back(new Manager("Razan Usman", 100, 1000, "CS", 2000.0));
    employees.push_back(new Engineer("Armaghan Atiq", 420, 1000, 10.0, 5));
    employees.push_back(new Salesperson("Abdullah Ijaz", 666, 1000, 10, 10));

    // Calculate and display salaries
    for (Employee* e : employees) {
        cout << "Name: " << e->name << endl;
        cout << "ID: " << e->id << endl;
        cout << "Base Salary: $" << e->baseSalary << endl;
        cout << "Total Salary: $" << e->calculateSalary() <<endl;
        cout << endl;
    }

    // Free memory
    for (Employee* e : employees) {
        delete employee;
    }
    return 0;
}
```

Output:

```
Name: Razan Usman
ID: 100
Base Salary: $1000
Total Salary: $30000

Name: Armaghan Atiq
ID: 420
Base Salary: $1000
Total Salary: $ 1015.0

Name: Abdullah Ijaz
ID: 666
Base Salary: $1000
Total Salary: $ 1020
```

Zainab Saad
23L-0646

```
class Employee {
protected :
    string name;
    int id;
    ~~basesalar~~ double basesalary;
public :
    Employee ( string n= " ", int i ~~id~~= 0 , double b = ~~1000~~ 0.0 ){
        name = n;
        id = i;
        basesalary = b;
    }
    virtual void print(){
        cout << name << endl << id << ~~bas~~ endl << basesalary;
    }
    virtual void calculatesalary() = 0;
    Employee(); // default constructor made in the end.
    virtual ~ Employee(); // in the end
};
→ class Manager : public Employee {
    private:
    string department ;
    double bonus ;
    public:
    Manager (string n= " ", int i= 0, double b= ~~1000~~ 0.0, string d= " ", double s = 0.0) :
                                Employee( n, i, b) {
        department= d;
        bonus = s;
    }
    Manager(); // default made in the end
```

```
void calculatesalary(){
        cout << ~~basesalary +~~ "Salary of manager";
double d ~~cout <<~~ = basesalary + bonus;
        cout << d;
}

void print() {
        cout << name << endl << id << endl << basesalary << endl << department << bon
}
};
```

→ class Engineer : public Employee {

```
    protected :
        double rate;
        int numProjects;
    public :
        Engineer(){
            rate = 0.0;
            numProjects = 0;
        }
        Engineer(string n="", int i=0, double b=0.0, double r=0.0, int num=0):
                                    Employee(n, i, b){
            rate = r;
            numProjects = num;
        }
        void calculatesalary(){
            double d = rate × numProjects;
            cout << "salary" << d+ basesalary;
        }
```

```
void Print() {
    cout<< name << id << basesalary << rate << numprojects;
}
~ Engineer() {
cout<< "destructor for engineer called";
}
};
```

→ class SalesPerson : public Employee {

```
    private;
        double salesachieved;
        double commissionrate;
    public:
        SalesPerson() {
            salesachieved = 0.0;
            commissionrate = 0.0;
        }
        SalesPerson(string n=" ", int i=0, double b=0.0, double s=0.0, double c=0.0) :
                                    Employee(n, i, b) {
            salesachieved = s;
            commissionrate = c;
        }
        void calculatepay() {
            double c = saleachieved x commission rate;
            cout << "salary of salesperson :" << c + basesalary;
        }
        void Print() {
            cout<< name << id << basesalary << salesachieved << commissionrate;
        }
        ~ salesperson(); //in the end.
};
```

Foundation for Advancement of Science & Technology

```
Employee :: Employee () {        // default
    name = " ";
    id = 0;
    basesalary = 0.0;
}

Manager :: Manager () {  // default
    department = " ";
    bonus = 0.0;
}
Employee :: ~Employee () {
    cout << "destructor called for employee";
}
SalesPerson :: ~SalesPerson () {
    cout << "destructor for salesperson";
}
Manager :: ~Manager () {
    cout << "destructor for manager";
}
```