Assignment 05

Q1 :

```
#include <iostream>
#include <string.h>
using namespace std;
class Employee
{
private:
    int id;
    char name[20];
    double salary;
    // Costructor

public:
    Employee()
    {
        // cout << "\nDefault Costructor called\n";
        this->id = 0;
        strcpy(this->name, "No Name");
        this->salary = 0;
    }

    Employee(char *name, int id, float salary)
    {
        // cout << "\nParameterized Costructor for Employee called";
        strcpy(this->name, name);
        this->id = id;
        this->salary = salary;
    }

    // Setters
    void setId(int Id)
    {
        this->id = Id;
    }
    void setName(char *name)
    {
        strcpy(this->name, name);
    }
    void setSalary(float salary)
    {
        this->salary = salary;
    }
    // Getters
    int getId()
    {
        return this->id;
    }
    char *getName()
    {
```

```cpp
        return this->name;
    }
    float getSalary()
    {
        return this->salary;
    }

    // Display
    virtual void display()
    {
        cout << "\nId : " << this->id << "\tName :" << this->name << "\t Salary :" <<
this->salary;
    }
    // Calculate salry
    virtual double calSal()
    {
        return salary;
    }
    virtual void sendSalary()
    {
        double totalSalary = this->calSal();
        cout << "Salary sent to employee." << endl
             << "Amount : " << totalSalary << endl;
    }
};

class SalesManager : public Employee
{
private:
    float incentive;
    int target;

public:
    // Costructor
    SalesManager()
    {
        // cout << "\nDefault Costructor called\n";

        this->incentive = 0;
        this->target = 0;
    }

    SalesManager(char *name, int id, float salary, float incentive, int target) :
Employee(name, id, salary)
    {
        // cout << "\nParameterized Costructor for SalesManager called";

        this->incentive = incentive;
        this->target = target;
    }

public:
    // Setters
    void setIncentive(float incentive)
    {
        this->incentive = incentive;
    }
    void setTarget(int target)
    {
        this->target = target;
    }

    // Getters

    float getIncentive()
    {
        return this->incentive;
    }
    int getTarget()
    {
        return this->target;
    }

    // Display
    void display()
    {
        Employee::display();
        cout << "\tIncentive : " << this->incentive << "\tTarget : " << this->target;
    }

    // Calculate salary
    double calSal()
    {
        return this->getSalary() + incentive;
    }
    // Send Salary
    void sendSalary()
    {
        double totalSalary = this->calSal();
        cout << "Salary sent to employee." << endl
             << "Amount : " << totalSalary << endl;
    }
};

class Admin : public Employee
{
    // id,name,salary,allowence
private:
    float allowence;

public:
    // Construuctor

    Admin()
    {
        // cout << "\nDefault Costructor called Admin\n";
        this->allowence = 00;
    }

    Admin(char *name, int id, float salary, float allowence) : Employee(name, id, salary)
    {
        // cout << "\nParameterized Costructor for Admin called";

        this->allowence = allowence;
    }

    // Setters

    void setAllowence(float allowence)
    {
        this->allowence = allowence;
    }

    // Display
    void display()
    {
        Employee::display();
        cout << "\tAllowence :" << this->allowence;
    }

    // getters

    float getAllowence()
    {
        return this->allowence;
    }
    double calSal()
    {
        return this->getSalary() + allowence;
    }
    // Send Salary
    void sendSalary()
    {
        double totalSalary = this->calSal();
        cout << "Salary sent to employee." << endl
             << "Amount : " << totalSalary << endl;
    }
};

class HR : public Employee
{
private:
    float commission;

public:
    // Costructor
    HR()
    {
```

```cpp
        // cout << "\nDefault Costructor called HR\n";
        this->commission = 0;
    }
    HR(char *name, int id, float salary, float commission) : Employee(name, id, salary)
    {
        // cout << "\nParameterized Costructor for HR called";

        this->commission = commission;
    }

    // Setters

    void setCommission(float commission)
    {
        this->commission = commission;
    }

    // Getters

    float getCommission()
    {
        return this->commission;
    }

    // Display
    void display()
    {
        Employee::display();
        cout << "\tCommission : " << this->commission;
    }

    double calSal()
    {
        return this->getSalary() + commission;
    } // Send Salary
    void sendSalary()
    {
        double totalSalary = this->calSal();
        cout << "Salary sent to employee." << endl
             << "Amount : " << totalSalary << endl;
    }
};

class AreaSalesManager : public SalesManager
{
private:
    char location[20];

public:
    AreaSalesManager()
    {
        // cout << "\nDefault Costructor for AreaSalesManager called";
        strcpy(location, "Not Given");
    }
    AreaSalesManager(char *name, int id, float salary, float incentive, int target, char
*location) : SalesManager(name, id, salary, incentive, target)
    {
        // cout << "\nParameterized Costructor for AreaSalesManager called";
        strcpy(this->location, location);
    }

    void display()
    {
        SalesManager::display();
        cout << "\tLocation :" << this->location;
    } // Send Salary
    void sendSalary()
    {
        double totalSalary = this->calSal();
        cout << "Salary sent to employee." << endl
             << "Amount : " << totalSalary << endl;
    }
};

int main()
{
    Employee *employee[5];
    employee[0] = new SalesManager("Bhagvat", 123, 500000, 1200, 22);
    employee[1] = new AreaSalesManager("Bhagvat", 123, 690000, 1200, 2, "Pune");
    employee[2] = new HR("Pinto", 124, 560000, 345);
    employee[3] = new Admin("Teja", 122, 780000, 3233);
    cout <<
"\n.......................................................................................
...........................\n";
    for (int i = 0; i < 4; i++)
    {
        employee[i]->display();
        cout << "\n\nTotal Salary :" << employee[i]->calSal() << endl;
        cout << endl;
        employee[i]->sendSalary();
        cout <<
"\n.......................................................................................
...........................\n";
    }

    // cout << "\n\nSales Manager Data :\n";
    // SalesManager s1("Bhagvat", 123, 690000, 1200, 2);
    // s1.display();

    // cout << "\n\nArea Sales Manager Data :\n";
    // AreaSalesManager As1("Bhagvat", 123, 690000, 1200, 2, "Pune");
    // As1.display();

    // cout << "\n\nHR Data :\n";
    // HR hr("Pinto", 124, 560000, 345);
    // hr.display();
    // cout << "\n\nAdmin Data :\n";
    // Admin admin("Teja", 122, 780000, 3233);
    // admin.display();

    return 1;
}
```
Output:

```
PS D:\Fullstack-Java-FirstBit-Solutions>  & 'c:\Users\bhagv\.vscode\--dbgExe=C:\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'
.............................................................................................................

Id : 123        Name :Bhagvat   Salary :500000 Incentive : 1200       Target : 22
Total Salary :501200
Salary sent to employee.
Amount : 501200

.............................................................................................................

Id : 123        Name :Bhagvat   Salary :690000 Incentive : 1200       Target : 2     Location :Pune
Total Salary :691200
Salary sent to employee.
Amount : 691200

.............................................................................................................

Id : 124        Name :Pinto     Salary :560000 Commission : 345
Total Salary :560345
Salary sent to employee.
Amount : 560345

.............................................................................................................

Id : 122        Name :Teja      Salary :780000 Allowence :3233
Total Salary :783233
Salary sent to employee.
Amount : 783233

.............................................................................................................
PS D:\Fullstack-Java-FirstBit-Solutions>
```

Q2:

```
#include <iostream>
#include <string.h>
using namespace std;

struct Shapes
{
    char shapeName[20];
    virtual float calculateArea()
    {
        cout << "\nShapes CalculateArea called\n";
```

```cpp
        return 0;
    }
    // void draw()
    virtual void draw()
    {
        cout << "\nShape Draw called\n";
    }
};

struct Vartul : public Shapes
{
private:
    float radious;

public:
    // Constructor
    Vartul(float red)
    {
        this->radious = red;
        strcpy(this->shapeName, "Vartul");
    }
    Vartul()
    {
        strcpy(this->shapeName, "Vartul");
        this->radious = 0;
    }

    // Setter
    void setRadious(float radious) { this->radious = radious; }
    // Getter
    float getRadious() { return this->radious; }
    // Area of Circle
    float calculateArea() override
    {
        return 3.14 * (this->radious * this->radious);
    }
    void draw()
    {
        cout << "\nVartul Draw called\n";
    }
};

struct Trikon : public Shapes
{
private:
    float base;
    float height;

public:
    // Constructor
    Trikon(float base, float height)
    {
        strcpy(this->shapeName, "Trikon");
        this->base = base;
        this->height = height;
    }
    Trikon()
    {
        strcpy(this->shapeName, "Trikon");
        this->base = 0;
        this->height = 0;
    }
    // Setter
    void setBase(float base) { this->base = base; }
    void setHeight(float height) { this->height = height; }
    // Getter
    float getBase() { return this->base; }
    float getHeight() { return this->height; }

    // Area of Trikon
    float calculateArea() override
    {
        return (0.5) * this->base * this->height;
    }
    virtual void draw()
    {
        cout << "\nTrikon Draw called\n";
    }
};

struct Aayat : public Shapes
{
private:
    float lambi;
    float width;

public:
    // Constructor
    Aayat(float lambi, float width)
    {
        strcpy(this->shapeName, "Aayat");
        this->lambi = lambi;
        this->width = width;
    }
    Aayat()
    {
        strcpy(this->shapeName, "Aayat");
        this->lambi = 0;
        this->width = 0;
    }
    // Setter
    void setWidth(float width) { this->width = width; }
    void setLambi(float lambi) { this->lambi = lambi; }
    // getter
    float getWidth() { return this->width; }
    float getLambi() { return this->lambi; }
    // Area of rectangle
    float calculateArea() override
    {
        return this->lambi * this->width;
    }
    virtual void draw()
    {
        cout << "\nAayat Draw called\n";
    }
};

struct Chauras : public Shapes
{
private:
    float baju;

public:
    // Constructor
    Chauras(float baju)
    {
        this->baju = baju;
        strcpy(this->shapeName, "Chauras");
    }
    Chauras()
    {
        this->baju = 0;
        strcpy(this->shapeName, "Chauras");
    }
    // Setter
    void setBaju(float baju) { this->baju = baju; }
    // Getter
    float getBaju() { return this->baju; }

    // Area Of square
    float calculateArea() override
    {
        return this->baju * this->baju;
    }
    virtual void draw()
    {
        cout << "\nChauras Draw called\n";
    }
};

int main()
{
    Shapes *shape[5];
    shape[0] = new Trikon(12, 10);
    shape[1] = new Vartul(9);
    shape[2] = new Aayat(10, 12);
    shape[3] = new Chauras(12);
    for (int i = 0; i < 4; i++)
    {
        cout << "\nArea Of Shape " << shape[i]->shapeName << " : " << shape[i]-
>calculateArea();
        // cout << "\n.....................................\n";
        shape[i]->draw();
        cout << "\n.....................................\n";
    }

    // Trikon trikon(12, 32);
    // Vartul vartul(9);
    // Aayat aayat(6, 8);
    // Chauras chauras(10);

    // shape = &trikon;
    // cout << "\nArea Of Trikon : " << shape->calculateArea();

    // shape = &vartul;
    // cout << "\nArea Of Vartul : " << shape->calculateArea();

    // shape = &aayat;
    // cout << "\nArea Of Aayat : " << shape->calculateArea();

    // shape = &chauras;
    // cout << "\nArea Of Chauras : " << shape->calculateArea();

    return 0;
}
```

Output:

```
PS D:\Fullstack-Java-FirstBit-Solutions>  & 'c:\Users\bhagv\.vscode\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'
Area Of Shape Trikon : 60
Trikon Draw called
.....................................
Area Of Shape Vartul : 254.34
Vartul Draw called
.....................................
Area Of Shape Aayat : 120
Aayat Draw called
.....................................
Area Of Shape Chauras : 144
Chauras Draw called
.....................................
PS D:\Fullstack-Java-FirstBit-Solutions>
```

3) Write a code to show polymorphic behavior where vehicle is base class and derived classes like bike, car, bus etc. Override the break function.

```
#include <iostream>
#include <string.h>
using namespace std;

struct Vehicle
{
    virtual void start() { cout << "\nVehicle Start"; }
    virtual void stop() { cout << "\nVehicle Stop"; }
    virtual void breaking() { cout << "\nVehicle Break"; }
};

struct Car : public Vehicle
{
    void start() { cout << "\nCar Start"; }
    void breaking() { cout << "\nCar Break"; }
};

struct Truck : public Vehicle
{
    void start() { cout << "\nTruck Start"; }
    void breaking() { cout << "\nTruck Break"; }
};

struct Bike : Vehicle
{
    void start() { cout << "\nBike Start"; }
    void stop() { cout << "\nBike Stop"; }
};

int main()
{
    Car car;
    Truck truck;
    Bike bike;

    // car.start();
    // truck.start();
    // bike.start();

    // car.stop();
    // truck.stop();
    // bike.stop();

    // car.breaking();
    // truck.breaking();
    // bike.breaking();

    Vehicle *v;
```

```cpp
    v = &car;
    v->breaking();
    v->start();
    v->stop();

    v = &truck;
    v->breaking();
    v->start();
    v->stop();

    v = &bike;
    v->breaking();
    v->start();
    v->stop();

    return 0;
}
```

Output:
```
PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment05\output> & .\'q3Vehicle.exe'

Car Break
Car Start
Vehicle Stop
Truck Break
Truck Start
Vehicle Stop
Vehicle Break
Bike Start
Bike Stop
PS D:\Fullstack-Java-FirstBit-Solutions\Basic-C-and-CPP\CPP\Assignments\Assignment05\output>
```

4) Write 2 more codes to show polymorphic behavior on your own.
4.1)

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

class Notification
{
public:
    // void send()
    virtual void send()
    {
        cout << "\nNotification send";
    }
};

class EmailNotification : public Notification
{
public:
    void send()
    {
        cout << "\nEmailNotification  send";
    }
};

class SMSNotification : public Notification
{
public:
    void send()
    {
        cout << "\nSMSNotification  send";
    }
};

class PushNotification : public Notification
{
public:
    void send()
    {
        cout << "\nPushNotification send";
    }
};

int main()
{
    Notification *Notifications[5];
    for (int i = 0; i < 5; i++)
    {
        if (i / 2 == 0)
        {
            Notifications[i] = new EmailNotification();
        }
        else if (i % 2 == 0)
        {
            Notifications[i] = new SMSNotification();
        }
        else
        {
            Notifications[i] = new PushNotification();
        }
    }

    for (int i = 0; i < 5; i++)
    {
        Notifications[i]->send();
    }

    return 0;
}
```

Output:

```
PS D:\Fullstack-Java-FirstBit-Solutions>  & 'c:\Users\bhagv\.vscode\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'

EmailNotification  send
EmailNotification  send
SMSNotification  send
PushNotification send
SMSNotification  send
PS D:\Fullstack-Java-FirstBit-Solutions>
```

4.2)

```
#include <iostream>

using namespace std;
class Sorter
{
public:
    // void  sort()
    virtual void sort()
    {
        cout << "\nSorter  sort";
    }
};

class QuickSort : public Sorter
{
public:
    void sort()
    {
        cout << "\nQuickSort   sort";
    }
};

class BubbleSort : public Sorter
{
public:
    void sort()
```

```cpp
    {
        cout << "\nBubbleSort   sort";
    }
};

class MergeSort : public Sorter
{
public:
    void sort()
    {
        cout << "\nMergeSort  sort";
    }
};
int main()
{
    Sorter *Sorters[10];
    for (int i = 0; i < 10; i++)
    {
        if (i / 2 == 0)
        {
            Sorters[i] = new QuickSort();
        }
        else if (i % 2 == 0)
        {
            Sorters[i] = new BubbleSort();
        }
        else
        {
            Sorters[i] = new MergeSort();
        }
    }

    for (int i = 0; i < 10; i++)
    {
        Sorters[i]->sort();
    }

    return 0;
}
```

Output:

```
PS D:\Fullstack-Java-FirstBit-Solutions>  & 'c:\Users\bhagv\.vscode\C:\TDM-GCC-64\bin\gdb.exe' '--interpreter=mi'
QuickSort   sort
QuickSort   sort
BubbleSort   sort
MergeSort  sort
BubbleSort   sort
MergeSort  sort
BubbleSort   sort
MergeSort  sort
BubbleSort   sort
MergeSort  sort
PS D:\Fullstack-Java-FirstBit-Solutions>
```