Name: Patel Manan Maheshkumar
Roll No: CE-111
PPS-II(Lab-5)

1. Family and Person classes

```
#include <iostream>
#include<string>
using namespace std;
class Person
{
  string name;
  int age;
public:
  Person(string name=" ", int age=0)
  {
    this->name=name;
    this->age=age;
  }
friend class Family;
};
class Family
{
  Person p[20];
  int members_count;
public:
  Family(int count=0)
  {
    this->members_count=count;
  }
  void add_member()
  {
    string name;
    int age;
    cin >> name >> age;

    if(members_count < 20){
      p[members_count].name=name;
      p[members_count].age=age;
      members_count++;
    }
  }
  void print()
  {
    for(int i=0;i<members_count;i++)
      cout << p[i].age << " " << p[i].name << endl;
```

```
    }
};
int main()
{
    int n;
    cin >> n;
    Family f;
    for(int i=0;i<n;i++)
        f.add_member();

    f.print();

    return 0;
}
```

2. Product class - constructor and copy constructor

```
#include <iostream>
#include<string>
#include<iomanip>
using namespace std;
class Product
{
    string name;
    double price;
public:
    Product(string name, double price)
    {
        this->name=name;
        this->price=price;
    }
    Product(const Product &p)
    {
        this->name=p.name;
        this->price=2*p.price;
    }
    void print()
    {
        cout << fixed << setprecision(2);
        cout << name << " " << price;
    }
};
int main()
{
    string name;
    double price;
```

```
    cin >> name >> price;
    Product p1(name,price);
    Product p2(p1);
    p2.print();
    return 0;
}
```

3. A shopping cart

```
#include<iostream>
#include<string>
using namespace std;
class Item
{
public:
    string name;
    int price;
    Item(string name="", int price=0)
    {
        this->name=name;
        this->price=price;
    }
friend class ShoppingCart;
};
class ShoppingCart
{
    Item I[10];
    int count=0;
    int q[10]={0};
public:
    void add_item(Item it,int quantity)
    {
        int i=0,j=quantity,flag=1;

        for(;i<count;i++)
        {
            if(I[i].name==it.name)
            {
                q[i]+=j;
                flag=0;
            }
        }
        if(flag)
            {
                I[count].name=it.name;
                I[count].price=it.price;
                this->q[count]=j;
```

```
            count++;
        }
    }
    int calculate_bill()
    {
        int total_price=0;

        for(int i=0;i<count;i++)
            total_price+=l[i].price*q[i];

        return total_price;
    }
    void print()
    {
        for(int i=0;i<count;i++)
            cout << endl << l[i].name << " " << q[i];
    }
};
int main()
{
    Item items[10];
    int input_items_count, price;
    string name;
    cin >> input_items_count;
    for(int i = 0; i < input_items_count; i++) {
        getchar();
        cin >> name >> price;
        items[i] = Item(name, price);
    }

    ShoppingCart sc;
    int purchase_entries, quantity;
    cin >> purchase_entries;
    for(int i = 0; i < purchase_entries; i++) {
        getchar();
        cin >> name >> quantity;
        int j;
        for(j = 0; j < input_items_count; j++) {
            if(items[j].name == name)
                break;
        }
        sc.add_item(items[j], quantity);
    }

    cout << sc.calculate_bill();
    sc.print();
```

```
  return 0;
}
```

4. A Bus on a Route: profit or loss?

```
#include<iostream>
#include<string>
#include<iomanip>
using namespace std;
class Route;
class Bus
{
  string bus_id;
  string fuel_type;
  double mileage_per_litre;
  int max_passengers;

public:
  static double petrol_price_per_litre;
  static double diesel_price_per_litre;

  Bus(string bus_id, string fuel_type, double mileage_per_litre, int max_passengers)
  {
    this->bus_id=bus_id;
    this->fuel_type=fuel_type;
    this->mileage_per_litre=mileage_per_litre;
    this->max_passengers=max_passengers;
  }

  static void change_petrol_price(double petrol);
  static void change_diesel_price(double diesel);

friend double calculate_profit(Bus &bus,Route &route);
};
void Bus::change_petrol_price(double petrol)
  {
    Bus::petrol_price_per_litre=petrol;
  }
void Bus::change_diesel_price(double diesel)
  {
    Bus::diesel_price_per_litre=diesel;
  }
class Route
{
  string route_id;
  string source;
  string destination;
```

```
  double distance;
  int fare_per_passenger;

public:

  Route (string route_id, string source, string destination, double distance, int
fare_per_passenger)
  {
    this->route_id=route_id;
    this->source=source;
    this->destination=destination;
    this->distance=distance;
    this->fare_per_passenger=fare_per_passenger;
  }

friend double calculate_profit(Bus &bus,Route &route);
};

double Bus::petrol_price_per_litre=80.88;
double Bus::diesel_price_per_litre=75.77;

double calculate_profit(Bus &bus,Route &route)
{
  if(bus.fuel_type!="petrol" && bus.fuel_type!="diesel")
    return 0;
  double fuel_price=0.0;

  if(bus.fuel_type=="petrol")
    fuel_price=Bus::petrol_price_per_litre;
  if(bus.fuel_type=="diesel")
    fuel_price=Bus::diesel_price_per_litre;
  double profit=(route.fare_per_passenger * bus.max_passengers) - (route.distance /
bus.mileage_per_litre * fuel_price);
  return profit;
}
int main()
{
  static double petrol_price_per_litre, diesel_price_per_litre;
  cin >> petrol_price_per_litre >> diesel_price_per_litre;
  Bus::change_petrol_price(petrol_price_per_litre);
  Bus::change_diesel_price(diesel_price_per_litre);
  // cout << Bus::petrol_price_per_litre << " " << Bus::diesel_price_per_litre << endl;
  string bus_id, fuel_type;
  double mileage_per_litre;
  int max_passengers;
  cin >> bus_id >> fuel_type >> mileage_per_litre >> max_passengers;
  Bus bus(bus_id, fuel_type, mileage_per_litre, max_passengers);
```

```
    getchar(); // Removing newline from input buffer

    string route_id, source, destination;
    double distance;
    int fare_per_passenger;
    cin >> route_id >> source >> destination >> distance >> fare_per_passenger;
    Route route(route_id, source, destination, distance, fare_per_passenger);
    cout << std::fixed << std::setprecision(2) << calculate_profit(bus, route);

    return 0;
}
```

5. A small Company

```
#include<iostream>
#include<string>
using namespace std;
class Company;
class Employee
{
   string employee_id;
   string employee_name;
   string designation;
   int salary;
public:
   Employee(string employee_id="",string employee_name="",string designation="",int
salary=0)
   {
      this->employee_id=employee_id;
      this->employee_name=employee_name;
      this->designation=designation;
      this->salary=salary;
   }
   void increment(int salary)
   {
      this->salary+=salary;
   }
   void print()
   {
      cout << employee_id << " " << employee_name << " " << designation << " " << salary
<< endl;
   }
friend class Company;
};
class Company
{
   string name;
```

```
  Employee e[10];
  int i1=0;
public:
  Company(string name)
  {
    this->name=name;
  }
  void add_employee(Employee e1)
  {
    for(int i=0;i<10;i++)
      if(e[i].employee_id==e1.employee_id)
       return;

    e[i1]=e1;
    i1++;
  }
  void increase_salary(string employee_id,int increment_amount)
  {
    for(int i=0;i<10;i++)
    {
      if(e[i].employee_id==employee_id)
        e[i].increment(increment_amount);
    }
  }
  void print()
  {
    for(int i=0;i<i1;i++)
      e[i].print();
  }
};
int main() {
  string company_name;
  cin >> company_name;
  Company cmp("MyCompany");

  int n;
  cin >> n;
  string employee_id, employee_name, designation;
  int salary;
  Employee emp;
  for(int i = 0; i < n; i++){
    getchar(); // Removing newline from input buffer
    cin >> employee_id >> employee_name >> designation >> salary;
    emp = Employee(employee_id, employee_name, designation, salary);
    cmp.add_employee(emp);
  }
```

```
    int m = 0, increment_amount;
    cin >> m;
    for(int i = 0; i < m; i++) {
        getchar(); // Removing newline from input buffer
        cin >> employee_id >> increment_amount;
        cmp.increase_salary(employee_id, increment_amount);
    }
    cmp.print();
    return 0;
}
```