Name: Patel Manan Maheshkumar
Roll No: CE-111
PPS-2(Lab-8)

1. Vehicle Movement

```
#include<iostream>
#include<string>
#include<iomanip>
#include<cmath>

using std::cin; using
std::cout; using std::endl;
using std::string; using
std::fixed;
using std::setprecision;

class Point { private:
float x, y; public:
   Point(float x=0.0,float y=0.0)
   {
      this->x=x;
      this->y=y;
   }
   void move_north(float u){ y+=u; }
void move_south(float u){ y-=u; }    void
move_west(float u){ x-=u; }    void
move_east(float u){ x+=u; }    float
getx(){ return x;}    float gety(){ return y;}
void print() const {
      cout << fixed << setprecision(2) << x << " " << y << endl;
   }
};

class Vehicle { private:
   float calculate_displacement() const {
Point start(starting_position);       Point
end(current_position);       float x1=end.getx() -
start.getx();       float y1=end.gety() - start.gety();
float displacment = sqrt(x1*x1 + y1*y1);
return displacment;
   }
```

```
   const Point starting_position;     Point
current_position;     float
distance_covered; public:
   Vehicle(Point start) : starting_position(start), current_position(start),
distance_covered(0.0) {}     void move_north(float u){
current_position.move_north(u);        distance_covered+=u;
   }
   void move_south(float u){        current_position.move_south(u);
distance_covered+=u;
   }
   void move_west(float u){        current_position.move_west(u);
distance_covered+=u;
   }
   void move_east(float u){        current_position.move_east(u);
      distance_covered+=u;
   }
   void print() const {        cout << "Starting
position: ";
starting_position.print();        cout <<
"current_position: ";
current_position.print();        cout << "Distance
covered: " << fixed << setprecision(2) <<
distance_covered
<< endl;         cout << "Displacement: " << fixed <<
setprecision(2) <<
calculate_displacement() << endl;
   }
};

int main() {     float initial_x,
initial_y;     cin >> initial_x >>
initial_y;
   Point starting_position(initial_x, initial_y);     Vehicle
vehicle(starting_position);     int
number_of_moves;     char direction;
   float units;
   cin >> number_of_moves;     for(int i = 0; i <
number_of_moves; i++) {
      getchar();
      cin >> direction >> units;
switch(direction) {           case 'N':
```

```
            vehicle.move_north(units);              break;
case 'S':               vehicle.move_south(units);
break;          case 'W':
            vehicle.move_west(units);           break;
case 'E':
            vehicle.move_east(units);
break;
      }
   }
   vehicle.print();
}
```

2. Company Hierarchy

```
#include<iostream>
#include<string> using namespace
std;
class Person
{
   short int age;
protected:    string
gender;    string name;
public:
   Person(string name="",string gender="other",short int age=0)
   {
      this->name=name;       this->gender=gender;
      this->age=age;
   }
   void print()
   {
      cout << name << " " << gender << " " << age << " ";
   }
};
class Employee : public Person
{
   int yearly_salary;    protected:
   int employee_id;    static int
count;    string
employee_since;    string
designation;    string
department;    public:
   Employee(string name,string gender,short int age,string
```

```
department,string designation,string employee_since,int yearly_salary)  :
Person(name,gender,age)
   {
      this->yearly_salary=yearly_salary;        this-
>employee_since=employee_since;        this->designation=designation;
this->department=department;                this->employee_id  =  count;
count++;

   }
   void print()
   {
     cout << "E" << employee_id << " " ;       Person::print();
     cout << department << " " << designation << " " << employee_since
<< " " << yearly_salary << endl;

   }
   void increment(float p)
   {
      float p1=p/100;
     yearly_salary+=(p1*yearly_salary);
   }
   string getD(){        return
designation;
   }
   void setD(string D){
      designation = D;
   }
   int getS(){
      return yearly_salary;
   }
};
int Employee::count=1;
class Manager : public Employee
{
   protected:
   Employee* direct_report[10];     short int
direct_reports_count=0;     public:
   Manager(const Employee &e) : Employee(e) {}     void
print()     {
      Employee::print();
for (int i = 0; i < direct_reports_count - 1; i++) {
         for (int j = i + 1; j < direct_reports_count; j++) {              if
```

```
(direct_report[i]->getS() < direct_report[j]->getS()) {
Employee* temp = direct_report[i];                direct_report[i] =
direct_report[j];                direct_report[j] = temp;
            }
         }
      }
      for (int i = 0; i < direct_reports_count; i++) {          direct_report[i]->print();
      }
   }
   void add_direct_report(Employee* ptr)
   {
      direct_report[direct_reports_count]=ptr;       direct_reports_count++;
   }
   void change_designation(string existing_designation,string  new_designation)
   {
     for(int i=0;i<direct_reports_count;i++)
     {
        if(direct_report[i]->getD()==existing_designation)
direct_report[i]->setD(new_designation);
     }
   }
   void increase_salary(string designation,float p)
   {
      for(int i=0;i<direct_reports_count;i++)
     {
        if(direct_report[i]->getD()==designation)
           direct_report[i]->increment(p);
     }
   }
 };
int main() {
   string name, gender, department, designation, employee_since;    short age;
int yearly_salary;
   cin >> name >> gender >> age >> department >> designation;    cin >>
employee_since >> yearly_salary;
   Employee employee(name, gender, age, department, designation,
employee_since, yearly_salary);
   Manager manager(employee);

   int direct_reports_count = 0;
   cin >> direct_reports_count;
```

```
    Employee *employee_ptr;
    for(int i = 0; i < direct_reports_count; i++) {        getchar(); // removing
newline from input buffer        cin >> name >> gender >> age >> department
>> designation;        cin >> employee_since >> yearly_salary;
        employee_ptr = new Employee(name, gender, age, department,
                                    designation, employee_since,
                                    yearly_salary);
        manager.add_direct_report(employee_ptr);
    }

    getchar(); // Removing newline from input buffer     string
existing_designation, new_designation;     cin >> existing_designation
>> new_designation;
manager.change_designation(existing_designation,
new_designation);

    float increment_percentage;     cin >> designation
>> increment_percentage;
manager.increase_salary(designation, increment_percentage);

    manager.print();

    return 0;
}
```