OOP Sections
Lab # 3

Ex 1 : employee class.

Create a class called employee representing information in an employee payroll databases. The operations performed on employees should include ways to set and retrieve individual fields to construct and destroy an employee object and to calculate an employee‟s pay. The following should be public member functions:

- Employee constructor
- Getid return the employee id
- GetlastName return last name
- GetpayRate return payrate
- Gethours return hours worked
- CalcGrosspay return grosspay
- SetHours set the hours worked
- Setpayrate set the payrate
- The following should be friend functions:
- Operator << write an employee to a stream
- Operator >> read an employee from a stream

Include the following data members:

- string id;
- string lastName;
- Float payrate;
- Float hours;

Allow employee object to be initialized in two different ways using default arguments in the constructor.
For example:
Employee e1(1000);
Employee e2(1000, “johnson”, 30.5, 55.0);
Write a short program that tests the employee class by constructing, initializing, and displaying objects.
Calculate and display the weekly gross pay for a single employee.

Solution:

Main.cpp

```
#include <iostream>
#include "Employee.h"

using namespace std;

int main()
{
   Employee employee2("1000");
   Employee employee3("1000", "Jonson", 30.5, 55.0);
   int employeesNumber;
//Return GrossRate and PayRate formatted
   cout<<"1-";
   cout<<"Pay Rate for Jonsone: ";
   cout<<employee3.GetPayRate();
   cout<<"\n";
   cout<<"GrossRate for Jonsone: ";
   cout<<employee3.CalcGrossPay();
   cout<<"\n\n\n";
//Read Records from Input File and Write them in Output File Organized
//define input and output streams to read and write
   ifstream in("input.txt",ios::in);
   ofstream out("output.txt",ios::out);

//read number of employees
   in>>employeesNumber;
//for loop to read each record and
//write it in well formated in output file
   for(int i=0 ; i<employeesNumber ; i++)
   {
      in>>employee2;
      out<<employee2;
   }
   cout<<"2-Write Records in output file done";
   cout<<"\n\n\n";


   return 0;
}
```

Employee.h

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <fstream>
#include <iostream>

using namespace std;


const float PAYRATE_MIN = 0.0;
const float PAYRATE_MAX = 50.0;
const float HOURS_MIN = 0.0;
const float HOURS_MAX = 60.0;

class Employee
{
public:

   Employee(string eId);
   Employee(string eId , string name, float rate, float hours);
   ~Employee() {}
//member methods
   string GetId() ;
   string GetLastName() ;
   float GetPayRate() const ;
   float GetHours() const;
   void SetHours(float hours);
   void SetPayRate(float rate);
   float CalcGrossPay();

//operator overloading
   friend ifstream & operator>> (ifstream & in, Employee & e );
   friend ofstream & operator<< (ofstream & out, Employee & e);


protected:

private:
   string id ;
   string lastName;
   float payRate ;
   float hoursWorked;
};

#endif // EMPLOYEE_H
```

Employee.cpp

```
#include "Employee.h"

using namespace std;

Employee::Employee(string eId, string name, float rate, float hours)
{
   id = eId;
   lastName = name;
   payRate = rate;
   hoursWorked = hours;
}
Employee::Employee(string eId)
{
   id = eId;
   lastName = "";
   payRate = 0;
   hoursWorked = 0;
}
//methods that return the values of the student
//get employee id
string Employee::GetId()
{
   return id;
}

//get employee last name
string Employee::GetLastName()
{
   return lastName;
}
//get employee pay rate
float Employee::GetPayRate() const
{
   return payRate;
}
//get employee worked hours
float Employee::GetHours() const
{
   return hoursWorked ;
}
//two methods to set pay rate and hours
void Employee::SetPayRate(float rate)
{
   if(rate >= PAYRATE_MIN && rate <= PAYRATE_MAX)
      payRate = rate;
   else
```

```
      payRate = 0;
}
void Employee::SetHours(float hours)
{
   if(hours >= HOURS_MIN && hours <= HOURS_MAX)
      hoursWorked = hours;
   else
      hoursWorked = 0;
}
float Employee::CalcGrossPay()
{
   return payRate * hoursWorked ;
}

//operators overloading
// overloading operator >> to read the employee information from file
ifstream & operator>> (ifstream & in, Employee & e )
{
   float rate, hours;
   in>>e.id>>e.lastName>>rate>>hours;
//check if the rate or hours are not in the appropirate value
   if(rate >= PAYRATE_MIN && rate <= PAYRATE_MAX)
      e.payRate = rate;
   else
      e.payRate = 0;
   if(hours >= HOURS_MIN && hours <= HOURS_MAX)
      e.hoursWorked = hours;
   else
      e.hoursWorked = 0;
   return in;
}
// overloading operator << to write the employee information into file
ofstream & operator<< (ofstream & out, Employee & e)
{
   out<<"Id= "<<e.id<<" ";
   out<<"Last Name: "<<e.lastName<<" ";
   out<<"Pay Rate= "<<e.payRate<<" ";
   out<<"Hours Worked= "<<e.hoursWorked<<endl;
}
```