**Assignment V (Due April 9th)**

For this assignment, you'll be writing a class hierarchy using inheritance and polymorphism. The class hierarchy is deliberately being kept simple; a full project would elaborate these types much more completely.

A company Employee (Employee) class is the parent class. This must be an abstract class. The Employee class has a few simple data elements: A name identifier (a string), and a unique identifier the id (also a string). It has the usual getters and setters:

* string GetName() const;
* string GetId() const;
* void SetName (const string& NewName);
* void SetId(const string& NewId);

It also has 3 virtual functions (functions that may be implemented differently by child types). These are:

* GetType(). A const function that returns a string. For the Employee class, it returns the string “Employee”.

* ReadData(istream&). This method attempts to read the following data from the input stream: The name, which may contain spaces but is always on a single line; then the id. You will want to use getline() for this. The stream extraction (input) operator >> and getline() handle the end of line marker at the end of each line slightly differently. This method is responsible for reading the appropriate data and the ending newline, so the stream is ready to read the next line. This method returns true if all read operations were successful, false otherwise. (The istream.good() function will be useful.)

* WriteData(ostream&). This const method writes the employee’s name and his id, each on a line by itself immediately followed by a newline. It also returns true if successful, false otherwise; the ostream.good() method will be useful here.

The ReadData() and WriteData() methods should be pure virtual. Go ahead and implement them, they'll make implementing the child classes simpler, but this is an abstract class. (There are no general employees, only employees of a specific type. This class captures the data common to all company employees across all categories.)

The default value for the name is “0” (an empty string); the default value for the id is 0.

The Employee class has 2 child classes, both of which use public inheritance. The HourlyEmployee class has two extra data members. An hourly employee must have recorded whether or not he has a good experience to become a monthly employee. This is a one character value, either 'F' for does not have much experience or 'T' for he is an expert. The hourly employee also has a double, its his salary. In addition to a constructor with defaults for all values (which must include values for the Employee values as well, since a HourlyEmployee IS an Employee).

The HourlyEmployee class has the following public methods:

* char GetExpertEmployee() const;

// ExpertEmployee value must be either 'F' (for no experience) or 'T' for expert.

* double GetSalary() const;
* virtual string GetType() const;
* bool SetExpertEmployee(char NewExpert);
* bool SetSalary(double NewSalary);

// Salary must be double >= 1000 and <= 10000. Other values ignored. // returns true if parameter is accepted, returns false otherwise.

* virtual bool ReadData(istream& in);
* virtual bool WriteData(ostream& out) const;

Note the pre- and post-conditions for the getter and setter. Do not print any additional error messages. The ReadData() method begins by calling Employee::ReadData(in) to read the employees's basic information, and then reads value whether he is an expert or not, then the salary. Again, each will be on a line by themselves, and the ReadData() method is responsible for reading the entire line, including the newline marker at the end. Its return conditions are the same as for Employee::ReadData(). The WriteData() method begins by calling Employee::WriteData(out) to write the initial information, then writes the value if he is an expert or not, then the salary, each on a line by itself, immediately followed by a newline. Its return conditions are the same as for Employee::WriteData(). The GetType() method for a HourlyEmployee returns the string “Hourly Employee”.

The other child class of the Employee is a MonthlyEmployee. (We use camel-case for consistency with the other employee types). It needs a constructor which takes defaults for all parameters (including Employee's). In addition to its parent's data, a MonthlyEmployee has an int, its rank out of 10 (1 is very good, 10 is Not active employee). This has a default value of 10. In addition to the ReadData() and WriteData() methods, it should have a method called SetRank. This returns a bool, and takes an int as its parameter. The rank must be greater or equal to 1 and less than or equal to 10. If the parameter passed in within this range, rank is set to the new value and the method returns true; otherwise rank is unchanged and the method returns false. The MonthlyEmployee::GetType() method returns the string “Monthly Employee”. The ReadData() and WriteData() methods work similarly to the HourlyEmployees’ methods, calling the parent's methods first, then reading/writing the rank on a line by itself and returning the status of the stream.

This program does not derive any further classes from HourlyEmployee or MonthlyEmployee, but leave the GetType(), ReadData(), and WriteData() methods as virtual (these should not be pure virtual); when we design a class, we usually have an eye toward re-use and later extensions. We may want to define different types of HourlyEmployee or MonthlyEmployee for later programs, and this keeps the classes easy to extend.

The main program is very simple, focusing mostly on data-in, data-out. The key advantage to inheritance is that it allows polymorphism; we can write a line of code and put off until runtime the decision about which method the line of code will call.

Your main program will have an array of 100 pointers to Employees. Your program should set each to NULL at the beginning of the program, then read the provided data file. The format of the data file is quite simple: A one-character 'flag' indicating what the next employee is ('H' for HourlyEmployee, 'M' for MonthlyEmployee) on a line by itself, followed by the employee’s data. Your program will go through the data file, processing each employee one at a time. It will read the flag, instantiate the next unused pointer in the array as an employee of the appropriate subtype, and then call the ReadData() method for that employee.

Once you have read all the data, report (print to the screen) how many employees of each type are present (hourly and monthly employees). Then, demonstrate correct functioning of your WriteData() methods by producing a new output file with the same basic format as the original input file, except that the new file should list all the monthly-employees, then all the hourly employees. (It is not necessary to sort the employees in your array; you can iterate through the array of pointers twice to do this). A sample input file and output file are attached. Your program will be tested against a larger data file with the same format. You may assume there will be at least 1 employee in the input file, and no more than 100. Your program should NOT make any assumptions about how many of each type of employees will be present; some companies are mostly or entirely hourly employees, others mostly or entirely monthly employees. If you detect any employee of an unknown type (a flag value other than M, m, H, h), close the input file, print a brief error message, and end the program.

* The attached files are (Assignment PDF, sample input, sample output). • DO NOT add any extra classes or unnecessarily header or files
* Zip all your files and submit it on the Blackboard.
* Add some comments to illustrate your work.
* Submit the assignment before the deadline.