Massey University

Assignment 2 part 1

Deadline:	**Anytime before** Sunday 24 Sept 2017, time due 12:00 noon
Evaluation:	10 marks – which is 5% of your final grade
Late Submission*:*	5% per hour (or fraction of hour) it is late
Teams	The assignment can be done individually or in pairs (of at most 2 students)
Purpose:	Practice with C++ strings, inheritance and polymorphism.

Problem to solve:
In this assignment, you will design and implement various types of employees, from an imaginary *Marvel University,* as described below.

For this assignment, you will create various types of employees: *Professors (includs lecturers and tutors), Administrators, Teaching Assistants*. Professors and Administrators have annual contracts at fixed annual salaries. Each Administrator has a job title: librarian, receptionist, registrar, accountant, secretary and so on. Teaching Assistants are university PhD students who have hourly wages and work for a variable number of hours each month.

The payroll department is required by law to keep employees information consisting of the family name, given name(s) and Inland Revenue Department (IRD) number.

In addition, they need to keep track of :

- The annual salary for Professor and Administrators.
- The number of days of unpaid leave taken (the number of days the employee did not work hence the payment for those days should be deducted from salary) for Professors.
- The job title for Administrators
- The student ID number, hours worked per month and hourly pay for Teaching Assistants

A Professor's type is characterized by : IRD number, Family name, Given name, Annual salary, Number of days of leave without pay.

```
Example:   65-102-456   Steiger   Ludwig 79345.00  3
```

An Administrator's type is characterized by : IRD number, Family name, Given name, Annual salary, Job title

```
Example:   06-1342-0456   Bush GeorgeJr  59099.76  Librarian
```

while a Teaching Assistant's type is characterized by: IRD number, Family name, Given name, Student ID , Hours worked, Pay per hour

```
Example:  63-232-1456  Presley-Jackson  Priscilla 08001896 12  22.50
```

- Your task is to define and implement the following classes : a base class called **`Employee`**, two classes called **`SalaryEmp`** and **`TEmp`** derived from `Employee` and two other classes `Prof` and `Admin` both derived from `SalaryEmp`.
  `TEmp` represents a Teaching Assistant employee, `Prof` a Professor employee and `Admin` an Administrator employee.

An example of possible use of the types you should design and implement is shown in the program example in Figure 1. The output produced is presented in Figure 2. That output lists all employees and at the end prints the total payroll for the month.

Requirements:

The **`Employee`** base class should contain

- ✓ the following **protected** data members:
  `mFName` and `mGName`, string data members to store the family name and given name(s) of an Employee and a **private** data member `mIRD`, a string to store IRD numbers formatted like this: 60-456-809.
- ✓ at least the following member functions:
  - Constructor(s) and destructor
  - Mutators and accessors to set and get the value for every data member
  - one **virtual** functions: `print` that displays (in this order) the IRD number, family name and given name for employee objects; this print function should take a parameter of `ostream &` type
  - a **pure virtual** function **getNetMonthlyPay** that is used to compute the monthly net wages for an employee

The **`SalaryEmp`** derived class should contain

- ✓ the following data member as a protected data member:
  - `mAPay` (the annual salary --a double value)
- ✓ and the following member functions:
  - Constructor(s)
  - Mutators and accessors to set and get the value for the annual salary

The **`Prof`** class, derived from `SalaryEmp` class, should contain

- ✓ the following private data members:
  - `mDaysLeave` (an integer) to store the number of days taken in that month as unpaid leave
  - `const int mTAX = 28` that is the percent used for tax to be deducted from the salary
- ✓ and the following member functions:
  - Constructor(s)
  - Mutators and accessors
    - to set and get the value for `mDaysLeave`
    - override the **getNetMonthlyPay:** the wage of a Professor is computed by subtracting from the monthly[1] salary the deduction for unpaid leave (if any). To compute the net payment per month the tax should be deducted from the resulted number.
    - override `print` such that will display the IRD number, family name and given name(s), and the monthly net pay for a Professor

[1] For this assignment consider that a calendar year has 365 days (ignore leap years) and 12 months (i.e. to compute salary per month divide annual salary by 12)

The **`Admin`** class, derived from `SalaryEmp` class, should contain

- ✓ the following extra data member:
  - `mJob` (a string) to store the job title –you can assume when reading data from input file that the job title is one word only (like librarian, or financialOfficer,…)
  - `const int mTAX = 25` tax deducted –the tax to be deducted is 25% of salary
- ✓ and the following member functions:
  - Constructor(s)
  - Mutators and accessors:
    - to set and get the value for `mJob`
    - override the **getNetMonthlyPay**: the net payment of an administrator per month and is computed by deducting the tax from the salary
    - override `print` such that will display the IRD number, family name and given name(s), and the monthly net pay for an Administrator

The **`TEmp`** represents a Teacher Assistant employee and the class should be derived from Employee class. The TEmp class should contain

- ✓ the following extra data member:
  - `mID` (a string) to store the student ID number
  - `mHWorked` (an integer) to store the numbers of hours worked per month
  - `mHPay` (a double value) to store the hourly payment
  - `const int mTAX = 20` that is the tax deduction percent
- ✓ and the following member functions:
  - Constructor(s)
  - Mutators and accessors to
    - set and get the value for `mID, mHWorked, mHPay`
    - override the **getNetMonthlyPay :** the net pay of a Teacher Assistant is computed by deducting the tax from the product of the hourly pay and numbers of hours worked per month.
    - override `print` such that will display the IRD number, family name and given name(s), and the monthly net pay for a Teacher Assistant.

All declarations & definitions for the above classes must be written in a file called **a2p1.h**

Your a2p1.h file should be organized as follows:

a) comments with names of all authors of the code solution, assignment number, etc...
b) the definition of the function **info(),** that should display on standard device (monitor) the name(s) & ID(s) of the author(s) of the program.
c) the class Employee declaration followed by the implementation of all member functions of the Employee class as **non-inline functions**.
d) the class SalaryEmp declaration followed by the implementation of all its member functions as **non-inline functions**.
e) the class Prof declaration followed by the implementation of all its member functions, as **non-inline functions**.
f) the class Admin declaration followed by the implementation of all its member functions, as non-inline functions.
g) the class TEmp declaration followed by the implementation of all its member functions, **as non-inline functions**.

Note: Do not include any `main()` function in your a2p1.h file.

Hand-in: Submit a2p1.h electronically using STREAM**.**

Here is an example of how the types you define should behave.
For this main function:

1 A2P1main.cpp *

```cpp
#include <iostream>
#include "a2p1.h"
using namespace std;

int main(){
    const int SIZE=10;
    info();
    Employee* emps[SIZE];
    emps[0]=new TEmp();
    emps[1]=new TEmp("Wuz", "Chris","22-267-8012",  "12323405", 6, 25);
    emps[2]=new TEmp("Jackson", "Priscilla", "63-232-1456",  "08001896",  12, 23.50 );
    emps[3]=new Admin();
    emps[4]=new Admin("Bush", "GeorgeJr","06-1342-0456", 59099.76,  "Librarian");
    emps[5]=new Admin("Clever","Bob", "22-267-812", 79345.00, "Bussines Manager");
    emps[6]=new Prof();
    emps[7]=new Prof("Steiger", "Ludwig", "65-102-456", 79345.00, 3);
    emps[8]=new Prof("Flop", "Boby", "11-178-120", 69009.00, 0);
    emps[9]=new Prof("Daisy", "Ana", "34-567-812",67099.00, 3);
    double totalToPay=0.0;
    for (int i=0;i< SIZE;++i){
        totalToPay +=emps[i]->getNetMonthlyPay();
        emps[i]->print(cout);
        cout<<endl;
    }
    cout<<"\nTotal wages this month: $"<<setprecision(10)<<totalToPay;
    cout<<"\nThank you for using our program! Bye! "<<endl;
    return 0;
}
```

The output is:

A2P1main

```
159.234 Assignment 2P1
Duck D.    ID 345123
Calude E.  ID 987 123

TA00-0000-000   NO_TA      NO_TA           $0
22-267-8012     Wuz        Chris           $120
63-232-1456     Jackson    Priscilla       $225.6
ADM00-0000-000  NO_ADM     NO_ADM          $0
06-1342-0456    Bush       GeorgeJr        $3693.74
22-267-812      Clever     Bob             $4959.06
PR00-0000-000   NO_PR      NO_PR           $0
65-102-456      Steiger    Ludwig          $4291.15
11-178-120      Flop       Boby            $4140.54
34-567-812      Daisy      Ana             $3628.86

Total wages this month: $21058.94999
Thank you for using our program! Bye!
```

Please be aware we will perform more tests in order to check that your types implement all features required as mentioned above.

Miscellaneous:

1. The program must be your own work. Please be aware that you might be asked to explain to your lecturer how your program works. **If you cannot explain it, then it is not yours and you will get 0 marks for that assignment**. Attributing someone else's work as your own is plagiarism, and it is a violation of Massey University policy. We might file an official complaint against any student who we believe has committed plagiarism.
2. Marks will be allocated for: correctness, completeness, use of C++ constructs presented in class/tutorials, simple and clear solution, good documentation, and structured output display (on screen).
3. **Only const global variables are allowed.**
4. Using goto, **non-constant global variables** or C-like I/O constructs (i.e printf, fprintf, scanf, FILE*, etc) or STL tools (vector, lists, algorithms, etc.--not presented in class) is not allowed and it will be penalised.
5. Programs that **do not run or do not compile in the (Albany) labs, using gcc(SciTe), get 0 marks.**
6. Suspicious **similar solutions** will **all** get 0 marks-see also point 1 above.
7. Write YOUR ID NUMBER(S), and YOUR **FAMILY** NAME(S) first, assignment number, what the program does at the beginning of the file you send electronically and follow the documentation style recommended in tutorials and in the file " Documentation Style" on 159.234 Stream.
8. When working in pair, **send one solution file per team.**
9. The assignment will be previewed on Thursday lecture before the assignment is due.

If you have any questions about this assignment, please ask the lecturer before its due time!