Program Purpose

Write a complete C++ program to define and demonstrate use of classes. Specifically overloading operators.

Mandatory Instructions

This program will utilize two classes, NumDays and TimeOff. These classes will then be used in client source file.

Part I: NumDays Class

Create a class called *NumDays* which will represent number of work hours and automatically convert the hours to number of days. For example, 8 hours would be automatically converted to 1 day, 12 hours would be converted to 1.5 days, and 18 hours would be converted to 2.25 days.

Type the class declaration with data members and member function prototypes into a file named **numdays.h**. Place all member function definitions in a file named **numdays.cpp**.

Private data members:

- h*ours (int)*
- *days (double)*

Public member functions:

- Default constructor – sets *hours, days* to zero
- Overloaded constructor – accepts *hours* as parameter, automatically calculate number of equivalent days
- Mutators (setters) and accessors (getters) to set and get the *hours* and *days* values

```
// Mutators
void setHours(int);
void setDays(int);
// Accessors
int getHours() const;
double getDays() const;
```

- An overloaded - operator that subtracts two *NumDays* objects (operator- implemented as a member function)
- Ad overloaded ++ operator (prefix and postfix version) to increment number of *hours* stored in an object. The *days* should be automatically recalculated (operator++ implemented as member functions)

```
// Operator overloads
NumDays operator-(const NumDays &) const;
NumDays operator++();
NumDays operator++(int);
```

- An overloaded << operator that displays a *NumDays* object in the form **Days: d** (where d is the days private data member) "**:**" followed by **Hours: h** (where h is the hours private data member). See example below. This overload must be implemented as a non-member function.

```
Days: 2.5, Hours: 20
```

- An overloaded >> operator that accepts values for a *NumDays* object entered as a *double* value of days (e.g., 5.25) and stores the *days* and *hours* amounts correctly in the *NumDays* object's data members (assume 8 hours is one day). Input of 2.25 days would be stored as 18 hours.

```
// implemented as non-member functions
friend ostream& operator<<(ostream &out, const NumDays &);
```

Part II – TimeOff Class

Create a class called *TimeOff*. Place the class declaration with data members and member function prototypes into a file named **timeoff.h**. Place all member function definitions in a file named **timeoff.cpp**.

The purpose of the class is to track employee's sick leave, vacation time, and unpaid time off. It should have, as members, the following instances of the NumDays class described in part I.

Private data members:

- *name (string)*
- *id (string)*

- *maxSickDays* – A numDays object that records the maximum number of days of sick leave the employee may take.
- *sickTaken* – A NumDays object that records the number of days of sick leave the employee has already taken.

- *maxVacation* – A NumDays object that records the maximum number of days of paid vacation the employee may take.

  Input Validation: Company policy state that an employee may not accumulate more than 240 hours of paid vacation. The class should not allow the maxVacation object to store a value greater than this amount.

- *vacTaken* – A NumDays object that records the number of days of paid vacation the employee has already taken.

Public member functions:

- Default constructor – sets *name, id* to empty string, sets all NumDays objects to a default object with hours, days set to zero

- Overloaded constructor – accepts *name, id* of an employee. Sets all NumDays objects to a default object with hours, days set to zero.

- Mutators (setters) and accessors (getters) to set and get various values.

```
// Accessors
double getSickDays()const;
double getMaxSickDays()const;
double getVacDays()const;
double getMaxVacDays()const;

NumDays getVacBalance()const;      // These function must utilize operator- overload to
NumDays getSickBalace()const;      // subtract vacTaken from maxVacation and sickTaken from
                                   // maxSickDays and return resulting NumDays object


// Mutators
void setSickDays(int);             // Number of hours is passed to all these methods
void setMaxSickDays(int);          // Any additional hours must be added to already accumulated
void setMaxVacDays(int);           // hours
void setVacDays(int);

// Note: setVacDays should utilize ++ (postfix) operator to increment vacation hours taken.
//
void TimeOff::setVacDays(int hours)
{
       for(int i=0;i<hours;i++)
              vacTaken++;
}
```

- An overloaded >> operator that displays employee time off status (see sample output provided below). Use prototype given below.

```
friend ostream& operator<<(ostream &out, const TimeOff &);
```

Note: Class TimeOff is a client of NumDays, i.e., you will need to include its header and utilize its public interface to access private data members of NumDays objects.

Part III – Personnel Report

Modify given client program (see main() below) that uses an instance of the *TimeOff* class you designed in part II. Program should ask the user to enter employee name and id followed by the number of months an employee has worked for the company (some value between 1 and 6). It should then use the *TimeOff* object to calculate and display the employee's maximum number of sick leave and vacation days. Sample output:

Employees earn 12 hours of vacation leave and 8 hours of sick leave per month.

You MUST the code given below for your client code main(). Store your client code in prog6.cpp file.
Note the **TO DO** comment in the provided code. Fill in with your own code to capture custom user input and display new employee time off status.

```
int main()
{
	TimeOff frodo("12345", "Frodo Baggins");
	TimeOff someEmployee;

	int months = 4;

	// Set up available time off
	frodo.setMaxVacDays(months * 12);
	frodo.setMaxSickDays(months * 8);

	// Record time taken
	frodo.setVacDays(16);		// taken 2 days of vacation
	frodo.setSickDays(12);		// taken 12 hours of sick time

	// Show frodo's initial status
	cout << frodo << endl;

	// simulate 3 more months of work
	for (int i = 0; i < 3; i++)
	{
		frodo.setMaxVacDays(12);	// 12 hours earned each month
		frodo.setMaxSickDays(8);
	}
	frodo.setVacDays(24);			// 3 days of vacation taken

	// Show frodo's status
	cout << frodo << endl;

	//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
	// TODO: Ask user to provide another employee data
	// id, name, months of employment (between 1 and 6)
	// use someEmployee object to store/display this data
	//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

	system("pause");
	return 0;


}
```

DO NOT modify any code above this line

Expected output of the provided code… you custom employee input/status should display below.

```
Employee Name:  Frodo Baggins
------------------------------------------
Vacation Days/Hours Earned:     6.0/48
Vacation Days/Hours Taken:      2.0/16
Vacation Balance Remaining:     4.0/32
------------------------------------------
Sick Days/Hours Earned:         4.0/32
Sick Days/Hours Taken:          1.5/12
Sick Days Balance Remaining:    2.5/20
============================================

======== Eployee Time Off Report =========
Employee ID:    12345
Employee Name:  Frodo Baggins
------------------------------------------
Vacation Days/Hours Earned:    10.5/84
Vacation Days/Hours Taken:      5.0/40
Vacation Balance Remaining:     5.5/44
------------------------------------------
Sick Days/Hours Earned:         7.0/56
Sick Days/Hours Taken:          1.5/12
Sick Days Balance Remaining:    5.5/44
============================================
```

Program Documentation

Provide documentation header for your program/source files and for each function/method other than main().

```
// Program        6
// Description:   Program description
// Programmer:    Your name
// Class:         CS 2020, fall/spring 20xx
```

```
// Function:     Function/Method name
// Description:  Function purpose
// Programmer:   Your name
// Class:        CS 2020, fall/spring 20xx
// Parameters:   list and describe
// Returns:      describe
```

What to turn in?

Make sure you have completed your program according to the specifications given.
Make a photo of your program by typing in the following commands at the prompt:

```
$ photo prog6.log
$ ls –l
$
$ cat numdays.h
$ cat numdays.cpp
$ cat timeoff.h
$ cat timeoff.cpp
$ cat prog6.cpp
$ g++ prog6.cpp timeoff.cpp numdays.cpp
$ .\a.out
$ Ctrl+d
$ logout
```