# CS 304 Homework Assignment 1

Due: 11:59pm, Thursday, January 28th

This assignment is scored out of 64. It consists of 7 questions. The first 6 questions are to be completed on D2L as an online quiz. There is a programming question and you will need to put all your Java programs (`*.java`) as well as output files for this question in the folder named LastName_FirstName_CS304_HW1. Zip this folder, and submit it as one file to Desire2Learn. Do not hand in any printouts. Triple check your assignment before you submit. **If you submit multiple times, only your latest version will be graded and its timestamp will be used to determine whether a late penalty should be applied.**

## Short Answers

Complete Quiz 1 on D2L by the due date. You might see there is a time limit of 120 minutes on this quiz but it is not enforced so you can ignore it. Before you complete all questions, DO NOT submit! Doing so will prevent any further changes to the answers. You can save your answers for as many times as you want before submission.

## Programming Questions

(40pts) You are provided with a **`Date`** class and an incomplete **`EnhancedDate`** class. The **`EnhancedDate`** class inherits from the **`Date`** class with two more methods:

**`addDays(int numDays)`** – This method adds a certain number of days to the date represented by the current **`Date`** object. The changes should be left on the current object.

This task is done by first creating an object of the **`EnhancedDate`** class in the test driver with an initial date and then calling the **`addDays`** method on this object. For example, if the current **`EnhancedDate`** object, say **`currentDate`**, contains the date 1/1/2016, then after the **`addDays(20)`** method is called on **`currentDate`**, the date in it would be 1/21/2016 because 20 days has been added to the original date. This is not a simple addition because you need to take care of different months as well as **leap years**.

**`daysFrom(Date startDate)`** – This method computes the number of days elapsed from **`startDate`**, and returns an integer. If the **`startDate`** is earlier than the current day, the return should be positive, whereas if it is later than the current date, then this method should return a negative number. Hint: you can call the **`compareTo`** method provided in the **`Date`** class to check which of the two dates is older so that you can always count the days from the older date to the newer date. You may choose to use the above **`addDays`** method to simplify this implementation – you can keep adding a day to the older date until you bring it up to the newer date.

**Do NOT use any off-the-shelf implementations for the above methods. You are required to do it from scratch using basic date structures!**

## a. Completing the `EnhancedDate` class

Your task is to implement the **`addDays`** and the **`daysFrom`** methods in the **`EnhancedDate`** class to fulfill the required objectives. **Note that you are only supposed to touch these two methods. You are NOT allowed to create any other methods, instance variables, or make any changes to methods other than `addDays` and `daysFrom` or files other than "`EnhancedDate.java`". Points will be taken off if you fail to follow this rule.**

## b. Code Testing

You are provided with a test driver implemented by "**`TestDate.java`**" (**Do not make any changes to this file!**) so there is no need to write your own. You are also given a data file "**`testDates.dat`**" that contains 35 test dates. **This is a binary file and you will not be able to view its content using a text editor. We have not covered how to read and write this type of files so do not worry if you have no idea how it works. We will introduce the topic later this semester.**

Depending on your programming environment, the data file might need to be placed in different folders so that you test driver can read it. For jGRASP, you can leave the data file in the same folder as your java files. For NetBeans, you should place it in your project folder in which you see directories like **`build`**, **`nbproject`**, and **`src`**, etc.

Once you have completed the methods, you can run the test. You should create a plain text file named "**`output.txt`**", copy and paste the output (if your code crashes or does not compile, copy and paste the error messages) to this file and save it.

## Grading Rubric:

Code does not compile: -10
Code compiles but crashes when executed: -5
Changes were made to things other than the **`addDays`** and **`daysFrom`** methods: -5
Off-the-shelf implementation was used in the implementation: -35
Dates were hard-coded into any of the methods: -40
Has output file: 5
Code passes 35 test cases: 35 (each test case worth 1 point)

## Sample Output:

```
The current date is 1/30/2010 and 2 days are added.
The correct new date is 2/1/2010 and the one calculated by your
program is 2/1/2010.
Correct!

The current date is 2/28/2011 and 2 days are added.
The correct new date is 3/2/2011 and the one calculated by your
program is 3/2/2011.
Correct!

The current date is 2/28/2012 and 2 days are added.
```

```
The correct new date is 3/1/2012 and the one calculated by your
program is 3/1/2012.
Correct!

...

(1/1/2020).daysFrom(12/31/2020) is supposed to return -365 and your
return is -365
Correct!

(1/30/2009).daysFrom(2/2/2009) is supposed to return -3 and your
return is -3
Correct!

...

(5/19/2030).daysFrom(4/30/1902) is supposed to return 46771 and your
return is 46771
Correct!

(2/28/1904).daysFrom(7/19/1827) is supposed to return 27982 and your
return is 27982
Correct!

(6/8/1999).daysFrom(6/8/1999) is supposed to return 0 and your return
is 0
Correct!

...

Total test cases: 35
Correct: 35
Wrong: 0
```