CSC 210 - In-class Exercise (Chapter 3) -- Use C++ language / syntax to complete the following questions

1. **Convert this Java program into C++** // mark what change and to what, and also mark if any statement are unnecessary

```
//-------------------------------------------------
// Date.java    by Dale/Joyce/Weems    Chapter 1
// Supports date objects
// having year, month, and day attributes.
//-------------------------------------------------

public class Date
{
  private int year;
  private int month;
  private int day;
  public static final int MINYEAR = 1583;

  // Constructor
  public Date(int newMonth, int newDay, int newYear)
  {
    month = newMonth;
    day = newDay;
    year = newYear;
   }

  // Observers
  public int getYear()
  {
    return year;
  }

 public int getMonth()
  {
    return month;
  }

  public int getDay()
  {
    return day;
  }

  public int lilian()
  {
    // Returns the Lilian Day Number of this date.
    // Precondition: This Date is a valid date after
    // 10/14/1582.
    // Computes the number of days between 1/1/0 and
this date as if no calendar
    // reforms took place, then subtracts 578,100 so
that October 15, 1582 is day 1.

    final int subDays = 578100;  // number of calculated
days from 1/1/0 to 10/14/1582

    int numDays = 0;

    // Add days in years.
    numDays = year * 365;

    // Add days in the months.
    if (month <= 2)
      numDays = numDays + (month - 1) * 31;
    else
      numDays = numDays + ((month - 1) * 31) -
                  ((4 * (month-1) + 27) / 10);

   // Add days in the days.
    numDays = numDays + day;

    // Take care of leap years.
    numDays = numDays + (year / 4) - (year / 100) +
              (year / 400);

  // Handle special case of leap year but not yet leap
day.
    if (month < 3)
    {
        if ((year % 4) == 0)   numDays = numDays - 1;
        if ((year % 100) == 0) numDays = numDays + 1;
        if ((year % 400) == 0) numDays = numDays - 1;
      }

    // Subtract extra days up to 10/14/1582.
    numDays = numDays - subDays;
    return numDays;
  }

  public String toString()
  // Returns this date as a String.
  {
    return(month + "/" + day + "/" + year);
  }

}   // end Date class
```

```java
//-----------------------------------------------------------------------
// DaysBetween.java          by Dale/Joyce/Weems               Chapter 1
//
// Asks the user to enter two "modern" dates and then reports
// the number of days between the two dates.
//-----------------------------------------------------------------------

import java.util.Scanner;

public class DaysBetween
{
  public static void main(String[] args)
  {
    Scanner conIn = new Scanner(System.in);
    int day, month, year;

    System.out.println("Enter two 'modern' dates: month day year");
    System.out.println("For example, January 12, 1954, would be: 1 12 1954");
    System.out.println();
    System.out.println("Modern dates occur after " + Date.MINYEAR + ".");
    System.out.println();

    System.out.println("Enter the first date:");
    month = conIn.nextInt();
    day = conIn.nextInt();
    year = conIn.nextInt();
    Date date1 = new Date(month, day, year);

    System.out.println("Enter the second date:");
    month = conIn.nextInt();
    day = conIn.nextInt();
    year = conIn.nextInt();
    Date date2 = new Date(month, day, year);

    if ((date1.getYear() <= Date.MINYEAR)
       ||
       (date2.getYear() <= Date.MINYEAR))
      System.out.println("You entered a 'pre-modern' date.");
    else
    {
      System.out.println("The number of days between");
      System.out.print(date1);
      System.out.print(" and ");
      System.out.print(date2);
      System.out.print(" is ");
      System.out.println(Math.abs(date1.lilian() - date2.lilian()));
    }
  }
}
```

2. Using this class UML, write a C++ class and driver.

Person
- firstName: string - lastName: string - dayBD: int - monthBD: int - yearBD: int
+ Person (fName: string, lName: string, d:int, m:int, y:int) + getAge ( ): int + toString ( ) : string

getAge – will calculate age based on the yearBD
toString– will return a concatenation string of all attributes including age, which is calculated in getAge method

Driver: (1) Read each attribute value from the user;
(2) instantiate object using input values, and
(3) print on the screen all values including age of this person using toString method