CS222: Assignment 5 - Modular arithmetic

1. Submission deadline: Monday, 6 Feb at 3:00 pm.
2. Follow good coding practices to gain more marks.
3. No copying among the students or from the Internet or any other source.
4. The assignment can be submitted in groups of size ≤ 2.
5. Submit a `.cpp` file and a `.pdf` file.
6. Write the names and roll numbers of the students at the top of each file.
7. The files should be called

   `noModN_firstRollNumber_secondRollNumber.cpp`,

   `noModN_firstRollNumber_secondRollNumber.pdf`,
8. The pdf should contain the output obtained when the program was run.
9. In case you do not know about C++ templates, check `https://www.learncpp.com/cpp-tutorial/template-non-type-parameters/`.
10. For more information about the assignment: `https://stackoverflow.com/questions/66546257/in-c-can-we-create-a-class-for-each-integer`

---

1. (25 points) Recall the modular arithmetic that we studied in the class. Create a `class noModN`[1] using a C++ template with non-type parameter for N. It has a single private data which is an integer between 0 and $N-1$. This data should be initialised using a constructor that takes an arbitrary integer(may be positive or negative) and converts it to modulo N representation. Do not use `%` operator.

   For this, you will need to implement the `divide(int, int)` algorithm which returns the quotient and the remainder.

   Also create a default constructor.

   Overload the operators `+`, `-` (unary and binary[2]), `*` and `++`[3] for this class.

   Recall that in multiplication, you should go modulo N in each of the intermediate steps.

   Now, in the `main` procedure, take integer inputs `a, b, c` from the user and create objects of the `class noModN`. Output `(a+b)*c, -a, a - b, a++, ++a` in this exact sequence. The output should be clearly understandable.

   [If you are brave, you can output/input using `cout/cin`. I.e. by overloading the `<<` and `>>` operators. Write in the common assignment comment on Google classroom whether you have implemented it. No credits for this!]

[1] short for number modulo N.

[2] use the overloaded unary `-` and the overloaded `+` to implement the binary `-`

[3] Recall that there are two versions of the `++` operator: prefix `++` as in `++i` and postfix `++` as in `i++`