COS 126	Princeton University	Fall 2022

Programming Exam 2

Before the exam. Read this page of instructions before the exam begins. Do *not* start the exam (or read the next page) until instructed to do so.

Duration. Once the exam begins, you have 80 minutes to complete it.

Submission. Submit your solutions on *TigerFile* using the link from the *Exams* page. You may submit multiple times (but only the last version will be graded).

Check Submitted Files. You may click the *Check Submitted Files* button to receive partial feedback on your submission. We will attempt to provide this feature during the exam (but you should not rely upon it).

Grading. Your program will be graded primarily on correctness. Efficiency and clarity will also be considered. You will receive partial credit for a program that implements some of the required functionality. You will receive a substantial penalty for a program that does not compile.

Allowed resources. During the exam you may use only the following resources: course textbook, companion booksite, course website, course Ed, your course notes, and your code from the programming assignments. For example, you may not use *StackOverflow* or *Google*.

No collaboration or communication. Collaboration and communication during this exam are prohibited, except with course staff. A staff member will be outside the exam room to answer clarification question.

No electronic devices or software. Software and computational/communication devices are prohibited, except to the extent needed for taking this exam (such as a laptop, browser, and IntelliJ). For example, you must close all unnecessary applications and browser tabs; disable notifications; and *turn off your cell phone*.

Honor Code pledge. Write and sign the Honor Code pledge by typing the text below in the file `acknowledgments.txt`.

> *I pledge my honor that I will not violate the Honor Code during this examination.*

Electronically sign it by typing `/s/` followed by your name.

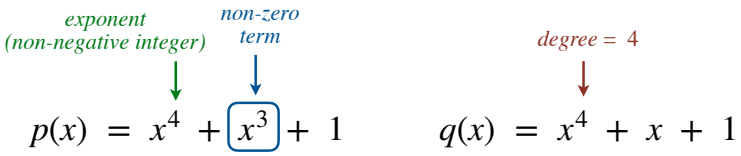
After the exam. Discussing or communicating the contents of this exam before solutions have been posted is a violation of the Honor Code.

Deliverables. You will submit two Java programs along with an `acknowledgments.txt` file:

1. A data type `BinaryPoly.java` for creating and manipulating *binary polynomials*.
2. A client program `BinaryPolyClient.java` that uses the `BinaryPoly` data type.

Parts 1 and 2 will be assessed independently; you may do Part 2 without completing Part 1.

Binary polynomials. A *binary polynomial* is a polynomial in which each coefficient is either 0 or 1, such as $p(x) = 1x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0$. When writing a polynomial, it's customary to omit terms whose coefficients are 0, leave off the 1 coefficients (since they are implicit), and substitute $x^0 = 1$:



The rules for manipulating binary polynomials are identical to those for manipulating integer polynomials except that all arithmetic is done modulo 2 (so that all coefficients remain 0 or 1).

- To *add* two binary polynomials, group the terms with the same exponent together into a single sum, remembering that $1 + 1 = 0$ and there are no "carries." For example, $r(x) = p(x) + q(x) = x^3 + x$.

$$\begin{array}{rr} & x^4 + x^3 \quad\quad + 1 \\ + & x^4 \quad\quad + x + 1 \\ \hline & x^3 \quad + x \end{array}$$

- The *degree* of a polynomial is the largest exponent of a non-zero term. For example, the degree of $q(x)$ is 4 and the degree of $r(x)$ is 3.

  By convention, the degree of the *zero polynomial* $z(x) = 0$ is -1.

Context. *Binary polynomials have many applications in computer science, including circuits, cryptograph, and error-correcting codes.*

Part 1 (19 points). Using the template file `BinaryPoly.java` provided in the project folder as a starting point, write a data type that implements the following API:

`public class BinaryPoly`	
`public BinaryPoly(int[] coefficients)`	*creates a polynomial from the coefficients*
`public String toString()`	*returns the string representation*
`public int degree()`	*returns the degree*
`public int coefAt(int i)`	*returns the coefficient (0 or 1) of* x^i
`public BinaryPoly plus(BinaryPoly that)`	*creates and returns a new polynomial that is the sum of the two polynomials*

Here is some more information about the required behavior:

- *Constructor.* The `coefficients[]` argument provides the coefficients of the terms for the polynomial, with `coefficients[i]` corresponding to the coefficient of x^i. For example, if the array is `[1, 0, 0, 1, 1]`, then the constructor should create the polynomial

$$1x^4 + 1x^3 + 0x^2 + 0x^1 + 1x^0.$$

  If the length of the array is 0, this corresponds to the zero polynomial; otherwise, you may assume that the last element in the array is 1.

- *String representation.* The `toString()` method should return a string that represents the polynomial, such as `"x^4 + x^3 + 1"`. Specifically, the string should include each non-zero term in the polynomial, in *descending order* by exponent, with adjacent terms separated by the string `" + "`. If the coefficient of x^i is 1, the term should appear as
  - `"x^i"` for $i \geq 2$
  - `"x"` for $i = 1$
  - `"1"` for $i = 0$

  If all coefficients are 0 (i.e., the zero polynomial), the `toString()` method should return `"0"`.

- *Valid arguments.* Don't worry about invalid arguments. You may assume the following:
  - No argument is `null`.
  - In the constructor, each element of `coefficients[]` is either `0` or `1`.
  - In `coefAt()`, the argument `i` is between 0 and the degree (inclusive).

  However, you should *not* assume that the two polynomials to be added have equal degree.

- *Efficiency.* You should strive for the following performance characteristics as a function of d, where d is the degree of the polynomial(s):
  - The `degree()` and `coefAt()` methods should take *constant* time.
  - The `plus()` and `toString()` methods should take *linear* time.

  Note that only a small number of points will be awarded for efficiency.
- *Unit testing (optional).* You may include a `main()` method for testing, such as this one:

```
public static void main(String[] args) {

    // p(x) = x^4 + x^3 + 1
    int[] coefficients1 = { 1, 0, 0, 1, 1 };
    BinaryPoly p = new BinaryPoly(coefficients1);
    StdOut.println(p);               // prints x^4 + x^3 + 1

    // q(x) = x^4 + x + 1
    int[] coefficients2 = { 1, 1, 0, 0, 1 };
    BinaryPoly q = new BinaryPoly(coefficients2);
    StdOut.println(q);               // prints x^4 + x + 1

    // z(x) = 0
    int[] coefficients3 = { };
    BinaryPoly z = new BinaryPoly(coefficients3);
    StdOut.println(z);               // prints 0

    // call instance methods
    StdOut.println(p.degree());      // prints 4
    StdOut.println(zero.degree());   // prints -1
    StdOut.println(p.coefAt(0));     // prints 1
    StdOut.println(p.coefAt(1));     // prints 0
    BinaryPoly r = p.plus(q);
    StdOut.println(r);               // prints x^3 + x
}
```

Part 2 (6 points). Write a client program `BinaryPolyClient.java` that takes a filename as a command-line argument; adds the binary polynomials described in the file; and prints the sum to standard output.

Use the `BinaryPoly` *data type to do the arithmetic (even if you did not implement it correctly).*

Input format. The input file contains a positive integer d (the degree of each polynomial), followed by the polynomials to add together, one per line. Each such line contains the $d + 1$ coefficients (0 or 1) of the polynomial, separated by whitespace, in descending order by exponent.

You may assume that the input is in the prescribed format and that it contains at least one polynomial.

Example. Here is a sample execution:

```
~/Desktop/poly> more poly3.txt
3  <------  degree of binary polynomials
1 1 1 1  <------  coefficients, in descending order by exponent
1 0 0 0  <------  x^3
1 1 0 1  <------  x^3 + x^2 + 1
1 0 0 1
1 1 1 1
1 0 1 0

~/Desktop/poly> javac-introcs BinaryPolyClient.java

~/Desktop/poly> java-introcs BinaryPolyClient poly3.txt
x^2 + x  <------  sum of binary polynomials in poly3.txt
                                                  |
                                                  v
                                                filename
```