Pre-test Guidelines

I. To qualify for the FPP track, you must have sufficient skill in the following general areas:
   1. Basic procedural programming and general problem solving
   2. Experience in at least one programming language
   3. See https://compro.miu.edu/sample-test/ for sample FPP pretests

II. For the MPP track, a deeper grasp of the the topics listed above is necessary, and the student should understand how to apply them in the context of the Java programming language. In addition, the student should have sufficient understanding of
   1. Data structures (lists and list nodes, stacks, queues, hash tables, trees)
   2. Recursion and iteration
   3. The object-oriented paradigm: working with objects, interfaces, inheritance, and polymorphism in the Java language

   For each of these topics, there will be one question on the MPP pretest that tests your understanding.

III. As a guide to the level of required experience in Java, essential topics are listed below. You can review these topics by reading *Core Java*, Volume 1, Horstmann and Cornell, 9th edition. Chapter references are given beside each of the topics below.

- the String, StringBuffer/StringBuilder classes (for creating and concatenating Strings) (chapter 3, Strings)

- Java methods: return values, parameter passing (chapter 4, see Mutator and Accessor Methods, and Private Methods for example)

- basics of classes and objects: class construction, object references, invoking a method on an object (chapter 4, Object Construction)

- Data Structures

  Be able to use these classes from the Java libraries
  - `ArrayList` (p. 658 and elsewhere)
  - `LinkedList` (p. 664)
  - Stacks and Queues
  - `HashMap` (p. 707-8)
  - Tree (`TreeMap`) (pp. 681, 691)

  Be able to create the following data structures on your own, *without* the help of the Java libraries:
  - `ArrayList` (p. 658 and elsewhere)
  - `LinkedList` (p. 664)
  - Stacks and Queues

  Note: Basic knowledge about how to use generics, like List<Integer>, HashMap<String, String>, will be assumed.

- Using the methods inherited from the Object class – in particular, you need to know how to override `toString` and `equals` as necessary.

- OO concepts: objects, interfaces, passing messages between objects, polymorphism.