A binary expression involves two operands (usually numbers, but we can use letters to represent them for this example) and operators; for arithmetic expressions in Java, the operators are: +, -, *, / and %. For logical expressions, the operators are && and ||, and for relational expressions, the operators are >, <, >=, <=, == and !=. Any of the above expressions can be represented by a **binary expression tree**. For example, the (infix) arithmetic expression:

\[
A + B * C * D - E
\]

can be represented by the tree:

```
     -
    / \
   +   E
  /   /
A =   D
  \
   /
  B C
```

A recursive algorithm can be developed that reads an expression in prefix form and builds its expression tree:

- if the term is an operand, use the data value to create a leaf node (left and right children are null)
- if the term is an operator, assign it as the data value of a node and create its left and right children

Your assignment is to:

- **Add a static method to the BTNode class that builds a binary expression tree from the prefix expression contained in a String argument:**

  ```java
  public static BTNode<E> buildBET(String prefix)
  ```

- **Using your method and other methods from the BTNode class, write a Java program that:**
  - Reads a string containing a well-formed prefix expression (may be arithmetic, relational, and/or logical) and creates a binary expression tree from this data
  - Prints the resulting tree using the print method in the BTNode class
  - Prints the expression represented by the binary expression tree in prefix, infix and postfix form
  - Repeats the steps above until the user signals a desire to quit
Grading Criteria for Program 4:

buildBET method (13 points):

- Differentiates between operands (numbers and letters) and operators (4) __________
- Recognizes arithmetic, relational, and logical operators (4) __________
- Builds tree correctly from given String (5) __________

Client program (12 points):

- Reads input from user and checks validity of prefix expression (3) __________
- Recovers from badly-formed expression without crashing (2) __________
- Calls buildBET and prints resulting tree using BTreeNode’s print method (2) __________
- Prints the expression in prefix and postfix form (3) __________
- Contains loop that runs until user wants to quit (2) __________