/\*\*

\* The Document class is used to store and modify the contents of a

\* text document.

\*

\* Operations are provided for reading, inserting, and deleting characters

\* in the document.

\* For convenience, cursor-relative operations are provided to make it easier

\* to edit a document in the style supported by a typical text editor.

\*

\* Domain:

\* Content: Sequence<char>, Document content

\* Length: Integer, Number of characters in the document (initially zero)

\* Cursor: Integer, Position in the character sequence where some

\* operations on the document occur (initially zero)

\*

\* @invariant length() >= 0

\* @invariant 0 <= cursor() <= length()

\*/

class Document {

/\*\*

\* Initializes empty document

\*

\* @pre None

\*

\* @post new length() = 0

\* @post new cursor() = 0

\*/

Document(){}

/\*\*

\* Returns the number of characters in the document

\*

\* @pre None

\*

\* @post Return value contains the number of characters in the document

\*/

int length() { return 0; }

/\*\*

\* Returns the specified number of characters, starting at the specified

\* position

\*

\* @param pos Position of the characters to be returned

\* @param count Number of characters to be returned

\*

\* @pre count <=length()

\* @pre pos <= length()

\* @pre count >=0

\* @pre pos + count <= length()

\*

\* @post string.length() == count

\* @post returns String

\* @post cursor remains at current pos

\*

\*

\*/

String read(int pos, int count) { return null; }

/\*\*

\* Inserts a character sequence into the document

\*

\* @param pos Position at which the character sequence should be inserted

\* @param s Character sequence to be inserted

\*

\* @pre s.length() >=0

\* @pre pos <= length()

\* @pre cursor at pos

\*

\* @post cursor() += s.length

\* @post length() += s.length

\*

\*

\*

\*/

void insert(int pos, String s) {}

/\*\*

\* Deletes a character sequence from the document.

\*

\* @param pos Starting position of the sequence that is being deleted

\* @param count Number of characters to be deleted

\*

\* @pre cursor() at pos

\* @pre pos <= length()

\* @pre pos + count <= length()

\*

\* @post cursor at pos

\* @post length() -= count

\*

\*

\*

\*/

void delete(int pos, int count) {}

/\*\*

\* Returns the current cursor position

\*

\* @pre None

\*

\* @post Return value contains the current cursor position

\*/

int cursor() { return 0; }

/\*\*

\* Sets the current cursor position

\*

\* @param pos New cursor position

\*

\* @pre 0 <= pos <= length()

\*

\* @post new cursor() = pos

\*/

void setCursor(int pos) {}

/\*\*

\* Inserts a single character into the document at the current cursor

\* position, and advances the cursor forward one position

\*

\* @param c Character to be inserted

\*

\* @pre none

\*

\* @post length() +=1

\* @post cursor() +=1

\*

\*

\*

\*

\*

\*/

void insert(char c) {}

/\*\*

\* Deletes the character at the current cursor position

\*

\* @pre cursor() <length

\*

\* @post cursor() remains at same place

\* @post length() -=1

\*

\*

\*

\*

\*

\*/

void delete() {}

/\*\*

\* Moves the cursor backward one position, and deletes the character at the new cursor position

\*

\* @pre cursor() >0

\*

\* @post cursor() -=1

\* @post length() -=1

\*

\*

\*

\*

\*

\*/

void deletePrevious() {}

/\*\*

\* Deletes the entire contents of the document and resets it to an empty state

\*

\*

\* @pre none

\*

\* @post length() =0

\* @post cursor() =0

\*

\*

\*

\*

\*/

void clear() {}

}