CSE 220 Lab 4 Pseudo Code
Team 26: Ashley Krueger, Emily Falkner, Savannah Puckett, and Scott Dexheimer
https://github.com/Thelandofsunshine/sunshineandrainbows

BTIterator.cpp:

- `void BTIterator::add(BinaryTreeNode** head, const char *nm, int line_num)`
  - Adds a binary tree with the specified head, name, and number of lines
- `void BTIterator::clear(BinaryTreeNode *n)`
  - Clears the specified binary tree
- `BinaryTreeNode *BTIterator::get_next(BinaryTreeNode *head, BinaryTreeNode *last)`
  - Gets the next element of of the binary tree with the specified head element and last element
- `BinaryTreeNode *BTIterator::find(BinaryTreeNode *head, const char *nm)`
  - Finds the element specified in the specified binary tree and returns a pointer to the node
- `BinaryTreeNode *BTIterator::find_parent(BinaryTreeNode *root, char *target)`
  - Given the root of the binary tree an the target, this method returns a pointer to the parent node
- `BinaryTreeNode *BTIterator::examin_children(BinaryTreeNode *root, char *target)`
  - This method looks at the children of a node given the root and the target and returns a pointer to the children
- `BinaryTreeNode *BTIterator::find_next(BinaryTreeNode *root, char *target)`
  - This method finds the next node in the binary tree given the root and the target and then returns a pointer to the node
- `BinaryTreeNode *BTIterator::get_leftmost(BinaryTreeNode *root)`
  - This method gets the leftmost node in the binary tree given the root and returns a pointer to this node

BinaryTreeNode.cpp

- `BinaryTreeNode::BinaryTreeNode(char *nm)`
  - One of the constructors for the BinaryTreeNode, takes just the name
- `BinaryTreeNode::BinaryTreeNode(char *nm, int ln)`
  - Another constructor for the BinaryTreeNode, takes both the name and the line
- `char *BinaryTreeNode::get_name()`
  - An accessor method, aka a getter method, that returns the name
- `LineNumNode *BinaryTreeNode::get_lines()`
  - An accessor method, aka a getter method, that returns the lines
- `BinaryTreeNode *BinaryTreeNode::get_left()`
  - An accessor method, aka a getter method, that returns the left item in the binary tree
- `BinaryTreeNode *BinaryTreeNode::get_right()`
  - An accessor method, aka a getter method, that returns the right item in the binary tree
- `void BinaryTreeNode::set_name(char *nm)`
  - A setter method, aka mutator method, that sets the name
- `void BinaryTreeNode::set_left(BinaryTreeNode *lft)`
  - A setter method, aka mutator method, that sets the left element in the binary tree
- `void BinaryTreeNode::set_right(BinaryTreeNode *rght)`
  - A setter method, aka mutator method, that sets the right element in the binary tree
- `void BinaryTreeNode::add_line(int num)`
  - A setter method, aka mutator method, that adds a line to the binary tree
- `void BinaryTreeNode::rem_lnn(LineNumNode *n)`
  - recursively removes the object's line list

    - calls itself in an if statement until all the items in the list are removed

LineNumNode.cpp

- `LineNumNode::LineNumNode(int num)`
    - The constructor for LineNumNode, takes just the number
- `int LineNumNode::get_number()`
    - A getter method, aka accessor method, that returns the integer number
- `LineNumNode *LineNumNode::get_next()`
    - A getter method, aka accessor method, that returns a pointer to the next LineNumNode
- `void LineNumNode::set_num(int num)`
    - A setter method, aka mutator method that sets the number
- `void LineNumNode::set_next(LineNumNode *nxt)`
    - A setter method, aka mutator method that sets the next node of the LineNumNode

Print.cpp

- `Print::Print(char source_name[], char date[])`
    - The constructor for print, takes both the name and the date
- `void Print::printLine(char line[])`
    - The printLine method prints the array of characters
- `void Print::printPageHeader()`
    - The printPageHeader method prints out the page header
- `void Print::printToken(Token *token)`
    - The printToken method prints out the formatted token that is given
- `void Print::printBT()`
    - The printBT method prints out the Binary Tree

Scanner.cpp

- `Scanner::Scanner(FILE *source_file, char source_name[], char date[], Print printer) : print(printer)`
    - The constructor for scanner, takes a file pointer, an array for the name and date, and an object of type print from the print class
- `bool Scanner::getSourceLine(char source_buffer[])`
    - Returns a Boolean value that says if there is a valid line or not
- `Token* Scanner::getToken()`
    - This is a getter method, aka accessor method, that returns a pointer to a token
- `char Scanner::getChar(char source_buffer[])`
    - This is a getter method, aka accessor method that returns a character
- `void Scanner::skipBlanks(char source_buffer[])`
    - This method scans through the characters in an array until there is a blank in order to skip the blanks
- `void Scanner::skipComment(char source_buffer[])`
    - This method scans through a line until it sees the symbols that denote a comment and then skips the text that follows
- `void Scanner::getWord(char *str, char *token_ptr, Token *tok)`
    - This method gets the next word, assigns a token to it, and then makes sure that the pointer now points after the word
- `void Scanner::getNumber(char *str, char *token_ptr, Token *tok)`
    - This method gets the next number, assigns a token to it, and then makes sure that the pointer now points after the number
- `void Scanner::getString(char *str, char *token_ptr, Token *tok)`
    - This method gets the next string, assigns a token to it, and then makes sure that the pointer now points after the string
- `void Scanner::getSpecial(char *str, char *token_ptr, Token *tok)`
    - This method gets the next special character, assigns a token to it, and then makes sure that the pointer now points after the special character

- `void Scanner::downshiftWord(char word[])`
    - This method takes a word in an array and then changes all of the letters in it to lowercase in order to use it to compare
- `bool Scanner::isReservedWord(char *str, Token *tok)`
    - This method takes in a pointer and then calls downshiftWord and compares the word to a predetermined list of reserved words

Token.cpp

- `Token::Token()`
    - The token object uses the default constructor
- `void Token::setCode(TokenCode newCode)`
    - This method sets the token code to something new for a token
- `TokenCode Token::getCode()`
    - This method is a getter method, aka an accessor method, that returns the token code for a token
- `void Token::setType(LiteralType newType)`
    - This method is a setter method, aka a mutator method, that sets the type for a token to a new type
- `LiteralType Token::getType()`
    - This method is a getter method, aka am accessor method, that returns the literal type for a token
- `void Token::setLiteral(int newInteger)`
    - This method is a setter method, aka a mutator method, that sets the literal for a token to a new integer value
- `int Token::getIntLiteral()`
    - This method is a getter method, aka an accessor method, that gets the integer value for the literal and returns it
- `void Token::setLiteral(float newReal)`
    - This method is a setter method, aka a mutator method, that sets the literal for a token to a new float value
- `float Token::getRealLiteral()`
    - This method is a getter method, aka an accessor method, that gets the float value for the literal and returns it
- `void Token::setLiteral(string newString)`
    - This method is a setter method, aka a mutator method, that sets the literal for a token to a new string
- `string Token::getStringLiteral()`
    - This method is a getter method, aka an accessor method, that gets the string for the literal and returns it
- `void Token::setTokenString(string s)`
    - This method is a setter method, aka a mutator method, that sets the token string for a token to a new string
- `string Token::getTokenString()`
    - This method is a getter method, aka an accessor method, that gets the string for the token string and returns it