CPS 501
ASSIGNMENT III
DOCUMENTATION

Presented By:

Thomas Sherk

Pengfei Guo

Vikas Gautam

BIG NUMBER CALCULATOR

Analysis:

- The program for big number calculator should be able to implement the computation for the big numbers using linked list (Singly Linked List or Doubly Linked List)
- We should be able to perform Addition, Subtraction, Multiplication, Greater than, Less than, and Equals operation for the given inputs.
- We should be able to design a GUI Scene Builder which should be able to perform all these operations listed below:
    1. TextField: receives input and output the result
    2. Label (on the right corner): display the current operator/comparison
    3. "+, - ,x": basic operator (add, subtract/multiply)
    4. ">,<,==": larger, smaller, equal comparison
    5. "Compute": Compute and display the result to the TextField "result"
    6. "Quit": quit the application
- Also, the label should be updated as the operator is selected
- The big number class should we implemented with the sub functions for all the other operators.

Issues:

- Addition: Storing values in linked list, taking care of decimal point, and performing the operation.
- Subtraction: Storing all the values in the linked list, taking care of the decimal point and performing the operation
- Multiplication: Storing the values in a linked list, taking care of the decimal point, performing multiplication, performing addition, and storing the result;
- Greater than, Less than, or equal than: Storing the values in a linked list, comparing values, adding and skipping the decimal point and comparing lists that are longer than each other.

Implementation:

- **Addition:**
    - Fill both linked lists using iteration
    - Find the decimal place of both lists, if there is none add to the end
    - Add zeroes to beginning of whole number and end of decimals to even out both lists
    - Reverse both lists
    - Remove the decimal place of both lists
    - Perform the addition process, if any combination of two integers is > 9, add 1 to next integer place and keep the remainder stored in current integer

- Reverse the final linked list
- Add back the decimal place if needed
- Set the result textfield to the answer

- **Multiplication:**
  - Fill two linked lists
  - Find decimal place, add one to end if needed
  - Calculate how many integers come after both decimals, sum them and store them in variable
  - Reverse both linkedlists
  - Remove the decimals
  - Perform multiplication by iterating the second linked list after multiplying the entire first linked list. Store value in seperate linked list seperated by + signs.
  - Perform addition by adding all multiplication linked lists
  - Reverse linked list
  - Add back digits
  - Set answer to textfield

- **Subtraction:**
    - Read from the text field and store as String value
    - Use a for loop to parse all char in the string to a LinkedList
    - Find decimal position
    - Align the two LinkedList based on the decimal position
    - Add zeros to the shorter linkedlist
    - Perform the minus operation
    - If the first digit is smaller than the second, add 10 to it and perform the operation
    - Return the result as a LinkedList
    - Convert it to a String to show on the GUI
- **Greater than:**
    - Read from the text field and store as String value
    - Use a for loop to parse all char in the string to a linkedlist
    - Find decimal position
    - Align the two linkedlist based on the decimal position
    - Add zeros to the shorter linkedlist
    - First check the decimal point: Decimal point position can tell us if it is greater
    - Then, if they have same decimal points, check each digit in the linkedlist.
    - Store the result, either “True” or “False” into a String

- **Less than:**
    - Read from the text field and store as String value
    - Use a for loop to parse all char in the string to a linkedlist
    - Find decimal position
    - Align the two linkedlist based on the decimal position
    - Add zeros to the shorter linkedlist
    - First check the decimal point: Decimal point position can tell us if it is smaller
    - Then, if they have same decimal points, check each digit in the linkedlist.
    - Store the result, either “True” or “False” into a String
- **Equal:**
    - Read from the text field and store as String value
    - Use a for loop to parse all char in the string to a linkedlist
    - Find decimal position
    - Align the two linkedlist based on the decimal position
    - Add zeros to the shorter linkedlist
    - Check each digit in the linkedlist is equal or not
    - Store the result, either “True” or “False” into a String
    - Show the String in GUI