Class: Huge Int *Specs* *by Gerardo Barcenas (CS 301-05)*

Private Aspects:

- Node Structure

  Each node is made of three parts
  - an int (digit)
  - a pointer towards the left (next_greater)
  - a pointer towards the right (next_lesser)
- Pointer towards Most Significant Digit (msd)
- Pointer towards Least Significant Digit (lsd)
- SignType value (sign)
  - Defines whether a Huge Int is positive or negative
- Number of digits in the Huge Int (numDigits)
- Void makeEmpty() method
  - Function: Clear LinkedList of all Nodes
    - Make sign = PLUS
    - While numDigits != 0
      - Make a new node temp
      - temp grab the node at msd
      - Make the msd->next_lesser the new msd
      - Free the temp
      - Subtract 1 from numDigits
      - Repeat;

Public Aspects:

- HugeInt() (Constructor)
  - Make both lsd and msd equal to nullptr
  - Make sign equal PLUS
  - Make numDigits equal 0
- ~HugeInt() (Destructor)
  - Use method makeEmpty()
  - Delete both lsd and msd
- bool operator<(const HugeInt &second) const
  - Check the sign of both HugeInt
    - If current is positive and second is negative: FALSE
    - If current is negative and second is positive: TRUE
    - If both are positives
      - If current has less digits: TRUE

                - If current has more digits: FALSE
                - If Both have the same amount of digits
                    - Go through each node and compare starting from the msd.
                        - Current found with a lesser value first: TRUE
                        - If both match or second has a lesser value: FALSE
            - If both are negative
                - If current has less digits: FALSE
                - If current has more digits: TRUE
                - If Both have the same amount of digits
                    - Go through each node and compare starting from the msd.
                        - Current found with a lesser value first: FALSE
                        - If both match or second has a lesser value: TRUE
- bool operator==(const HugeInt &second) const
    - If both signs of HugeInts don't match: FALSE
    - If both don't have the same amount of digits: FALSE
    - Check each node until both pointer nodes don't match
        - If mismatch found: FALSE
        - If we've gone through all nodes no mismatch was found: TRUE
- HugeInt operator+(const HugeInt &second) const
    - If current (callingHugeInt) 's sign is positive and second's is negative
        - Treat it like a subtraction
        - Whoever's value is bigger decides the result's sign in the end.
    - If current (callingHugeInt) 's sign is negative and second's is positive
        - Treat it like a subtraction
        - Whoever's value is bigger decides the result's sign in the end.
    - If both signs match
        - Add the current pointer nodes values into ReplacementDigit
            - (add any carryover from the last addition)
            - (if a node doesn't exist, make it equal 0)
        - Get the carryover for the next digit by dividing by 10
        - Get the digit we'll use in the summation using modulus of 10
        - Keep repeating until both pointers are nullptrs and there's no carryover to the next digit
- HugeInt operator-(const HugeInt &second) const
    - If current (callingHugeInt) 's sign is positive and second's is negative
        - Treat it like a addition

        - The sign will always be positive
    - If current (callingHugeInt) 's sign is negative and second's is positive
        - Treat it like an addition
        - The result will be always negative
    - If both signs are negative
        - Make them both positive
        - Subtract them again but now as positives
    - If both signs are positive
        - Starting from the lsd of both HugeInt, subtract the digits
        - Make the bigger absolute value HugeInt on the top and the smaller on the bottom (negatives will be put back in the end)
        - *EX: 3 - 5 will look like "5 - 3" this to the computer, the subtraction happens, and then if the left (current's digit value: 3) is smaller then the right (second's digit value: 5), the result will return with a negative sign*
        - If the top number is smaller than the bottom number, add 10 to the top number and remove 1 from the next top digit.
        - Repeat until both pointers equal nullptr
- void operator=(const HugeInt &second)
    - Uses makeEmpty method to make the current HugeInt empty
    - Starting from second's lsd, go through each second'd nodes and use insertDigit() method with the parameter being the pointer node's digit value
- void insertDigit(int newDigit)
    - If numDigits equals 0
        - Make new node curr
        - Make curr's digit equal newDigit
        - Make curr's next_lesser and next_greater equal nullptr
        - Make both msd and lsd point to curr
    - If numDigits is greater or equal to 1
        - Make new node curr
        - Make node temp equal msd
        - Make curr's digit equal newDigit
        - Make curr's next_greater equal nullptr
        - Make curr's next_lesser equal msd
        - Make temp's next_greater equal curr
        - Make msd point to curr
- void printDigits() const
    - If sign equal MINUS, print "-"
    - If numDigits equals 0, print error: empty list
    - If numDigits is greater than or equal to 1

- Make new node temp equal msd
- For i < numDigits
    - cout temp’s digit
    - Make temp equal to temp’s next_lesser