**COMP2012H Assignment 2 Report**

### Author: WONG Yuk Chun (20419764)

**Task 1**

the link list is from most signiﬁcant digit to least signiﬁcant digit with sign as head.

## iostream operators

### :



**<<**

it is just similar to the passed.

to\_string

function, loop through the link list and print, then return ostream object

### :



**>>**

it is just calling the passed.

from\_string

function using the input string as parameter, then return istream object

## Comparison operators

### :

**a == b**

it is just looping through the two link list on the same time, once there is a diﬀerent character at the same corresponding position of the two lists, return false. If the list have diﬀerent length, return false. else it is true.

### :

**a != b**

it is inversing the result of

a==b

### :

**a > b**

if a is positive and b is negative, a of course bigger than b, true. if a is negative and b is positive, a of course smaller than b, false. if both are positive, if index of dot in a (from LHS) > that of b, a of course bigger than b, true. if index of dot in a (from LHS) < that of b, a of course smaller than b, false. if length of a > that of b, a of course bigger than b, true. if length of a < that of b, a of course smaller than b, false. the exception case is same length and same digit places are present in a and b check from most signiﬁcant digit to least digit by digit, once >  =>true <  =>false till the end no result=>they are the same=>false if both are negative, the return result is the inverse of result of both are positive.

### :

**a < b**

equivalent with

b > a

### :

**a >= b**

equivalent with inverse of

b > a

### :

**a <= b**

equivalent with inverse of

a > b

## Assignment operator

### a. :



**=**

remove all nodes in the link list, construct new link list by copying the value of the source link list.

## Arithmetic operators

### :

**a + b**

* 1. compare the absolute value of a and b, let a be the bigger and b be the smaller.
  2. align the dot position of a and b by adding zero at front and at the back, like a = XXXXXX.XX

b = X.XXXXX

will become

a = XXXXXX.XX000 b = 00000X.XXXXX

* 1. loop from least signiﬁcant ﬁgure to the most, if same sign, preform addition, else preform subtraction, increment or decrement LHS digit correspondingly when necessary.
  2. ﬁnally return the value with sign of a

### :

**a - b**

* 1. ﬂip the sign of copy of b, b', '+' to '-' or '-' to '+'
  2. return value of a+b'.

### :

**a \* b**

* 1. remove the dot of absolute value of a and b, namely a' and .



b'



0

0



\*

\*

* 1. if b' equals to 10, semi-product is appending if a' equals to 10, semi-product is appending

to a' , a base case for recurrence relationship of to b' , a base case for recurrence relationship of

if b' is one digit, semi-product (initially = 0) is the loop from ﬁrst digit of a', each time multiply by 10 and add the multiple of the digit and b.

else b' is more than one digit, semi-product is a' \* last digit of b' + 10\* a\*remaining digits of b'

* 1. add dot to the semi-product, location from right is the sum of number of decimal digits of a and b.
  2. set sign of the semi-product, if sign of a and b are the same, sign is +, if not sign is -, and now it is the ﬁnal product.

### :

**a / b**

* 1. if b equals to 0, throw division by zero error
  2. remove dot of absolute value of a and b, namely and .



a'



b'

* 1. calculate the quota for decimal points in the quotient, which is max of precision of a and b +1 (for rounding)
  2. Evaluate max\_d, which is the maximum number of zero can be appended to b' to make it just smaller than a'
  3. begin the recurrence relation:
     1. a' > b' no more quota end with NULL have quota do a' \* 10 /b and drop one quota, and if it is the ﬁrst time to use quota, append .



* + 1. else

evaluate number of times a' can be subtracted by and let it be d.

returned number is d coming with resultant digits of  which this / sign is calling the recurrence function with max\_d decrease by 1.

### :

**a ^ b**

* 1. if b<0, it is with precision of max of that of a and b, and round oﬀ accordingly.

a/abs(b)

* 1. if b equals 0, result is 1
  2. if b equals 1, result is a
  3. if b equals 2, result is a\*a
  4. else
     1. turn to a bit-string, set result as 1.
     2. for each digit in , if is 1,result multiply by 
     3. return result

### increment and decrement

: add 1 to this and return this

++a

: make a copy of a, add 1 to this, return the copy

a++

: minus 1 from this and return this



--a

: make a copy of a, minus 1 from this, return the copy



a--

# Task 2

The link lists will have dummy heads.

## 1

3 global node pointers in total:

1. temp: main accessing pointer
2. p: temp's parent node
3. M: pointing to the  node

temp traverse the link list from  data node to last data node, p start from the dummy head. when temp reach  node, record M as p

before temp reach node, remove the node which temp is pointing and insert it to where M is pointing, and continue with next node. After that, make temp pointing to pointee of p's next and continue.

when temp reach a NULL, print out the resultant link list

## 2

3 global node pointers in total:

1. m: main accessing pointer
2. h: temp's parent node
3. n: m's pointee's next node

temp traverse the link list from  data node to last data node, h start from the dummy head, n start from  data node.

if it sees m data equals to n data, delete n and make the next node as n, repeat this until m not equal to n and remove m. use p's next as m and p's next's next as n and continue.

when m or n reach NULL, print out the resultant link list.

## 3

2 global node pointers in total:

1. m: main accessing pointer
2. h: temp's parent node

temp traverse the link list from  data node to last data node, h start from the dummy head record current m's next as n, and n's next is k.

reverse m and n by making h's next as n, n's next as m, m's next as k. now it change from h->m->n->k to h->n->m->k

if k is actually null, terminate and output the link list.

take h as m and m as m's next and continue.

if m and m's next reach NULL, print out the resultant link list.