CS215 - Introduction to Program Design, Abstraction, and Problem Solving

Spring 2017

Programming Assignment #2

Note: Only Listing and Executions are required for this assignment. External Documentation is not required and should not be submitted for grading.

However, you will still need to write complete 3-part Specifications (including a Program Description, Program Preconditions, and Program Postconditions) to include in the Overall Header Comment of your program Listing. Also, you will need to develop a set of at least 6 proposed normal testcases, 6 proposed boundary testcases, and 6 proposed exception testcases to run in your program Executions.

Most humans have 10 fingers. Therefore, long ago, when arithmetic was first being invented, humans tended to count with their 10 fingers, and this led to the system of *decimal* (base 10) arithmetic. Decimal arithmetic uses 10 decimal digits, which are the digits from 0 through 9.

Most humans also have 10 toes. What if humans had invented an arithmetic system based on counting on their 10 fingers and 10 toes at the same time? Actually, some cultures did this (such as the ancient Mayans and Aztecs, and some current-day residents of Japan and Greenland), and this led to the development of *vigesimal* (base 20), arithmetic. Vigesimal arithmetic uses 20 vigesimal digits, which are the digits 0 through 9, as well as A ("ten"), B ("eleven"), C ("twelve"), D ("thirteen"), E ("fourteen"), F ("fifteen"), G ("sixteen"), H ("seventeen"), I ("eighteen"), and J ("nineteen").

(In the following examples, numbers which are spelled out, such as "thirty-six", refer to decimal numbers.)

Writing down a number in decimal that is less than its base ("ten") is easy -- just write down the corresponding decimal digit:

7 ("seven")
4 ("four")

Similarly, writing down a number in vigesimal that is less than its base ("twenty") is also easy -- just write down the corresponding vigesimal digit:

H ("seventeen")
6 ("six")

Writing down a number in decimal that is greater than or equivalent to its base ("ten") requires us to use more digits, and each digit's place will be "ten" times greater than the place on its right:

97 ("nine times ten, plus seven, which equals ninety-seven")
563 ("five times ten times ten, plus six times ten, plus three, which equals five hundred sixty-three")

Similarly, writing down a number in vigesimal that is greater than or equivalent to its base ("twenty") requires us to use more digits, and each digit's place will be "twenty" times greater than the place on its right:

3E ("three times twenty, plus fourteen, which equals seventy-four")
J4F ("nineteen times twenty times twenty, plus four times twenty, plus fifteen, which equals seven thousand six hundred ninety-five")

The rules for doing basic arithmetic, such as addition, in the vigesimal system are similar to those used in the decimal system: add digits from right to left, and don't forget to make carries.

For example, in decimal:

```
   4536
+ 69258
  -----
  73794
```

In other words, adding 6 ("six") and 8 ("eight") gives 14 ("fourteen"), which is at least as big as the base ("ten"), so write down the 4 ("four") and carry the 1 ("one") to the next column; adding 1 ("one") and 3 ("three") and 5 ("five") gives 9 ("nine"); adding 5 ("five") and 2 ("two") gives 7 ("seven"); adding 4 ("four") and 9 ("nine") gives 13 ("thirteen"), which is at least as big as the base ("ten"), so write down the 3 ("three") and carry the 1 ("one") to the next column; adding 1 ("one") and 0 ("zero") and 6 ("six") gives 7 ("seven").

Here's a different example in vigesimal:

```
  GE3H9
+  H5J3
  -----
  HB9GC
```

In other words, adding 9 ("nine") and 3 ("three") gives C ("twelve"); adding H ("seventeen") and J ("nineteen") gives 1G ("thirty-six"), which is at least as big as the base ("twenty"), so write down the G ("sixteen") and carry the 1 ("one") to the next column; adding 1 ("one") and 3 ("three") and 5 ("five") gives 9 ("nine"); adding E ("fourteen") and H ("seventeen") gives 1B ("thirty-one"), which is at least as big as the base ("twenty"), so write down the B ("eleven") and carry the 1 ("one") to the next column; adding 1 ("one") and G ("sixteen") and 0 ("zero") gives H ("seventeen").

Similar rules apply when multiplying two vigesimal numbers together.

By the way, the name "vigesimal" comes from the Latin word "vigesimus" which means "twentieth", and possibly also from the Sanskrit word "vimsatih" which means "twenty".

Design, write, and execute a C++ program to add and multiply vigesimal numbers, consisting of an arbitrary number of vigesimal digits.

The user should be able to work with an ongoing, "current" vigesimal number by repeatedly choosing from the following actions:

```
e   enter the current vigesimal number from the keyboard
a   add a new vigesimal number to the current vigesimal number
m   multiply a new vigesimal number by the current vigesimal number
h   help the user by displaying these choices
q   quit the program
```

Here are some requirements for your program:

- Your program should be able to handle whole vigesimal numbers, either positive or zero, containing any number of digits.

- When your program starts executing, it should automatically start with a current vigesimal number of 0.

- Each time before getting the user's choice, the program should display the current vigesimal number. The vigesimal number should be displayed in the conventional left-to-right order, and, as is customary, in groups of three digits (grouped from right-to-left) separated by commas. Furthermore, no more than 10 groups of three digits should be displayed on any single line of output -- if more space is needed, the program should skip to the next line, indent to the beginning of the vigesimal number plus an additional 2 to 4 spaces (whichever is needed to line up the groups vertically), and then continue displaying the vigesimal number.

- Whenever the user chooses to enter the current vigesimal number from the keyboard, he or she should be allowed to enter all of the vigesimal digits (0 through J) of that vigesimal number, without any embedded commas, in the conventional left-to-right order. The user should also be able to enter either uppercase or lowercase letters for any of the vigesimal digits A through J. In case the vigesimal number being entered is especially long, the user should be allowed to hit the "enter" key as many times as desired to go on to new, fresh lines. Finally, after all of the vigesimal digits have been entered, the user should signal the end of the vigesimal number by typing the special "#" character as the sentinel.

- Whenever the user chooses to add or multiply the current vigesimal number with a new vigesimal number, the program should correctly perform this calculation, making the resulting vigesimal number become the new current vigesimal number.

- All inputs given by the user to your program should undergo data validation to insure that they are correct in data type and range before being used. When processing an entered vigesimal number, your program should ignore any "whitespace" characters (blanks, tabs, or newlines), but it should warn the user whenever any other character (non-vigesimal digit and nonwhitespace) is seen, and then simply ignore that invalid character and continue processing the entered vigesimal number.

- You should write this program by extending the *LList* class that we are currently developing in our CS215 lecture. Each vigesimal number should be associated with a single *LList* object, with each vigesimal digit of that number stored as an element inside of a dynamically-created *listnode* object.

- Your program should perform the addition and multiplication calculations by directly analyzing and manipulating vigesimal digits; your program should not convert the given vigesimal numbers into any other base in order to perform the calculations. However, your program may convert individual vigesimal digits (0 through J) into decimal form for minor calculations, if you wish.

Here are some hints to help make your programming easier:

- You can assume that there will always be sufficient memory to allocate storage for new nodes in your linked lists.

- Your program does not need to handle negative or fractional vigesimal numbers.

- Store the vigesimal digits of each vigesimal number **backwards** in its associated *LList* object, that is to say, with the rightmost digits of the original vigesimal number in *listnode* objects near the head-end of the *LList*, and with the leftmost vigesimal digits in *listnode* objects near the tail-end of the *LList*. This will make addition and multiplication operations much easier and more efficient to code. (Keep in mind, however, that the order of the vigesimal digits in the vigesimal number would have to be temporarily reversed before displaying the number, in order to display the vigesimal number in the conventional left-to-right order.)

- Keep in mind that a multiplication of a vigesimal number can always be re-written in terms of a series related additions. For example, 4JG3 * 3H9 can be re-written as adding together 9 ("nine") copies of 4JG3, then H ("seventeen") copies of 4JG30, and finally 3 ("three") copies of 4JG300.

- You may find it helpful to understand the ASCII character set when working on this program. To learn more about the ASCII character set, go to http://www.google.com and search for ASCII, which will bring up several tens of millions of related pages. Take a look at a few of them, and in particular note the relationship of all of the lower-case letters to each other, and the relationship of all of the upper-case letters to each other.

- To convert a character into its associated numeric ASCII code, simply use that character value as if it were an *int*. For example, if *ch* is a variable of type *char* containing the character *'A'*, and *num* is a variable of type *int*, then:

  ```
  num = ch;
  ```

  would put the whole number *65* into *num* (since 65 is the ASCII code for the capital letter "A"). Similarly, a numeric ASCII code can be converted into its associated character value by using that number as if it were a *char*. So, continuing our earlier example:

  ```
  num += 2;
  ch = num;
  ```

  would put the character *'C'* into *ch* (since *num* was changed from *65* to *67*, and 67 is the ASCII code for the capital letter "C").

- As a way of tackling this program in stages, I would recommend first getting menu option "e" (entering vigesimal numbers from the keyboard) working. Then get menu option "a" (adding vigesimal numbers) working. Finally, get menu option "m" (multiplying vigesimal numbers) working. In each case, start with small vigesimal numbers for which you know the answers before trying out larger vigesimal numbers.

- Be very careful using pointers! Many programming errors can be traced to the incorrect use of pointers that are undefined (for example, that are uninitialized or are dangling). Be sure that each pointer you are using to access a *listnode* has directly or indirectly been initialized by using the *new* operator, and that it isn't *NULL*, and isn't dangling due to premature use of the *delete* operator. If you seem to be having trouble getting your pointers to work, a good debugging technique is to draw some sample *LList* objects by hand, and then trace through your code step-by-step looking for places where your pointers seem to go astray.

Be sure to follow the overall requirements for the Listing and Executions portion of programming assignments found in the "Programming Requirements" document posted on the CS215 Blackboard Course Shell. You'll need to run the set of required testcases I will make available shortly before the program listings and executions are due, as well as to develop and run your own thorough set of proposed testcases.

To complete this programming assignment, submit your Listing and Executions as a new, single PDF document attached to the appropriate submission link on the CS215 Blackboard Course Shell by the "Listing and Executions" due date stated there. (NOTE: You should create the PDF document for your Listing and Executions by using the CSTools “listexec” command on the District Unix computer system.)

External Documentation is not required for this assignment and should not be submitted for grading.

On the following pages is a sample of what your program execution should look like when completed.

Good luck!

```
Vigesimal Calculator, Version 1.0
(c) 2017, (your name)

Current vigesimal number is:    0

Command (h for help): h

Valid commands are:
   e   enter     enter the current vigesimal number from the keyboard
   a   add       add a new vigesimal number to the current vig. number
   m   multiply  multiply a new vigesimal number by the current vig. number
   h   help      show this help menu
   q   quit      quit the program

Current vigesimal number is:    0

Command (h for help): e

Enter a vigesimal number, followed by #: ge3h9#

Entering completed.

Current vigesimal number is:    GE,3H9

Command (h for help): a

Adding a new vigesimal number to the current vig. number.

Enter a vigesimal number, followed by #: h5j3#

New vigesimal number is:        H,5J3

Adding completed.

Current vigesimal number is:    HB,9GC

Command (h for help): m

Multiplying a new vigesimal number by the current vig. number.

Enter a vigesimal number, followed by #: 524EH60600GC7AJH9#

New vigesimal number is:        52,4EH,606,00G,C7A,JH9

Multiplying completed.

Current vigesimal number is:    4,9GF,9H9,1ED,5BD,7H4,3A5,ID8

Command (h for help): m

Multiplying a new vigesimal number by the current vig. number.

Enter a vigesimal number, followed by #: C5A4EG991D03068F
500B3084A7B9
2228E09
G55I9
      ?
```

```
FDID
BH8BB3
   AE1EI7
r68H

C2JA
E


#

Warning: invalid vigesimal digits seen: '?', 'r'; ignoring these

New vigesimal number is:        C,5A4,EG9,91D,030,68F,500,B30,84A,7B9,
                                  222,8E0,9G5,5I9,FDI,DBH,8BB,3AE,1EI,
                                  768,HC2,JAE

Multiplying completed.

Current vigesimal number is:    2F,2G9,9CB,EI7,722,359,H6C,DH6,64E,F1B,
                                   8HD,41E,HDH,989,2JA,JCG,J03,2G3,I62,
                                   IB9,I5I,DG9,I93,C72,A3H,5CF,IDG,J10,
                                   77C

Command (h for help): q

Finishing Vigesimal Calculator, version 1.0
```