ECGR 3180 Fall 2020
Dr. Mihail Cutitaru

Assignment 1
Assigned: September 29, 2020
Due: October 6, 2020 by 11:59PM in Canvas

1. **(40 pts)** The following problem involves the design of an unsigned hexadecimal/hex (base-16) to decimal (base-10) converter. The conversion can be done using the decimal equivalent of each hex digit multiplied by the weight of the digit. More information on the conversion and a calculator can be found at: https://www.binaryhexconverter.com/hex-to-decimal-converter All variable names will be given in double quotes in the prompt below to ensure all variables are named correctly.

Design a singly linked list where each node is a char named "value" and has a pointer named "link" pointing to the next node in the list. The link of the last node in the list points to NULL. Write code for the following:

- The struct described above, name it "digit".
- A class named "hex" to implement a singly-linked list that will contain any number of nodes, dependent on how many digits the user enters. The user will enter the hex digits one at a time and press the Enter key after every digit. When done, the user will enter '-' to signify the end of the linked list.
- When a new digit is entered, it will be stored as a node at the end of the linked list. Create a function named "addDigit" that will implement this functionality.
- The hex class will have a private variable named "count" to keep track of the number of nodes in the linked list and increase it every time a new node is added at the end of the list.
- The first node will have a pointer pointing to it named "head" that will never change its pointing. It points to the MOST SIGNIFICANT digit in the hexadecimal number. This will have an exponent value of count-1.
- A function named "convertToDecimal" that will iterate through all hex digits and convert the value to decimal and return it as an int. See the link at the top of this question for the procedure. The basic procedure is to multiply the hex digit (or its decimal equivalent if above 9) by the weight of each digit (this will be 16 "to the power of" the digit location). For example: hex number 2D9E in decimal is: 2*16^3 + D*16^2 + 9*16^1 + E*16^0 = 2*16^3 + 13*16^2 + 9*16^1 + 14*16^0 = 11678.
- The user may enter only valid hex digits (0 through F), no other input is allowed. If the value entered by the user is invalid, output an error message to the console and store the number entered so far without the invalid input.

```
class hex
{
      private:
            struct digit
            {
                  // YOUR CODE GOES HERE

            };

            // ANY ADDITIONAL CODE GOES HERE

      public:

            // MOST, IF NOT ALL, FUNCTIONS GO HERE

};

/*************************************************************/

#include <iostream>

using namespace std;

int main()
{
```

```
        // WRITE TEST CODE HERE, THE MORE TESTS THE BETTER.
        // ADD COMMENTS FOR EACH TEST CASE.

}
```

2. **(40 pts)** The following problem involves the design of function to merge multiple vectors using iterators. For more information and references to use the vector container, see http://www.cplusplus.com/reference/vector/vector/ or any other reference. This function will require the use of an iterator to traverse the vector.

- Write a function (named "mergeVectors") that will take in user input for a list of integers for each vector and output a merged vector to the console. The user will first enter a list of vectors as an array of vectors and then merging will occur. For an example of an array of vectors

-- The first value requested as user input is the number of vectors that will be entered. This number must be larger than 0. Include a checkpoint for this in your code and do no advance until proper input is given.

-- The code will then request for the user to enter the values in each vector. The values will be non-negative. If the user enters a negative value (such as -1 or more negative), the function will stop taking user data for the current vector and move to asking for data for the next vector in the array (entering a negative value essentially means the user is done entering data for the given vector.)

-- Populate the array with vectors (possibly of different sizes, depending on user input) using the approach described above.

- After the array of vectors has been populated with user data, iterate through every element in the array of vectors and output the elements to the console using the following algorithm:

-- The code will output the elements at index [0] for all vectors, then for index [1] if available, etc.

-- If a vector is shorter than the current index to be output, it will be skipped during output and the code will move to the next vector in the array, etc. See the examples below on how this merging will occur.

Example 1:
If the user has entered all of the vectors below as part of your code (this is just showing the vectors, not how they are stored), then the merging will be as follows:

v1 = [1, 2, 3, 4]
v2 = [5, 6, 7, 8]

Console output:
1, 5, 2, 6, 3, 7, 4, 8

Example 2: (spacing is added in this example for readability)

v1 = [1, 2, 3, 4, 9, 10]
v2 = [5, 6, 7, 8]
v3 = [9, 10, 11]

Console output:
1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 9, 10

Example 3: (spacing is added in this example for readability)

v1 = [1, 2, 3, 4, 9, 10, 11]

v2 = [5, 6, 7, 8]
v3 = [9, 10, 11]
v4 = [11, 12, 13, 14, 19, 20]

Console output:
1, 5, 9, 11, 2, 6, 10, 12, 3, 7, 11, 13, 4, 8, 14, 9, 19, 10, 20, 11

```
/*****************************************************************/

#include <iostream>

using namespace std;

void mergeVectors();

int main()
{
      //Testing your code
      mergeVectors();
}
/*****************************************************************/

// YOUR FUNCTION mergeVectors GOES HERE. IT WILL BE OF TYPE void AND TAKE
NO ARGUMENTS. ALL OPERATIONS WILL TAKE PLACE IN THIS FUNCTION.
```

3. (20 pts) For each one of the loops below, give a Big-Oh analysis of the running time. Show work/reasoning.

```
a) sum = 0;
   for (i = 0; i < n; ++i)
         ++sum;

b) sum = 0;
   for (i = 0; i < n; ++i)
         for (j = 0; j < n; j++)
               ++sum;

c) sum = 0;
   for (i = 0; i < n; ++i)
         for (j = 0; j < n*n; ++j)
               ++sum;

d) sum = 0;
   for (i = 0; i < n; ++i)
         for (j = 0; j < i; ++j)
               ++sum;

e) sum = 0;
   for (i = 0; i < n; ++i)
         for (j = 0; j < i * i; ++j)
               for (k = 0; k < j; ++k)
                     ++sum;
f) sum = 0;
   for (i = 1; i < n; ++i)
         for (j = 1; j < i * i; ++j)
               if (j % i == 0)
                     for (k = 0; k < j; ++k)
                           ++sum;
```