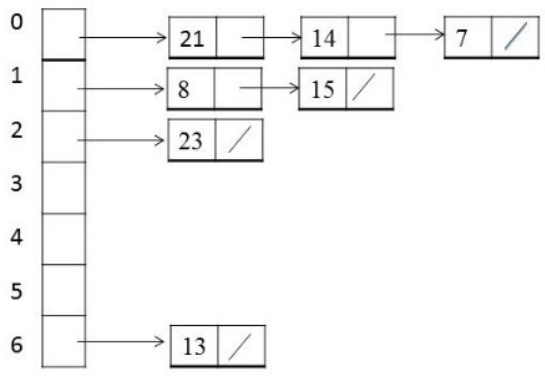
# Exercise 5 – Hashing

**Due by the end of your Week-8 laboratory class. (2 marks)**

This exercise is to be completed during your week 8 laboratory class. When you complete the exercise show your work to your lab tutor to have your work marked. The marking is based mainly on correct implementation and code readability. You should implement your code in one file (e.g. ex5.cpp, ex5.c, ex5.java). Make sure your program has a header comment block containing the name of the exercise, your name and your student login (e.g. jfk01). You may implement your solution in C, C++, Java or Python.

For this exercise, you are to implement a simple hash table. Your program should prompt for the name of an input file and the read and process the data contained in the file.

The file contains a sequence of integer values. Read them and construct a hash table using chaining. For this exercise, you should use linked lists implemented with dynamic data, as shown below



Your program should read each integer in turn and calculate its hash value using mod 100 as the hashing function. Thus, if the key is *k*, the hash value h(*k*) = *k* mod 100.

When you have finished calculate and print:
1. The number of empty entries in the hash table.
2. The length of the longest chain.

As usual, do not use STL or equivalent. Page 2 has pseudo code of some algorithms you might use.

When you are finished, test your program using the provided text file named “ex5.txt” and show your code and the output to your lab tutor to receive your mark. Also, submit your file via unix (banshee) using the submit command below.

```
$ submit –u login –c CSCI203 –a ex5 filename
```

where ‘***login***' is your UNIX login ID and ‘***filename***’ is the name of your file.

If you are unable to attend your lab class and demonstrate your work on time due to circumstances beyond your control (e.g. sickness), contact your lecturer  to request an extension.

# Pseudo Code

```
insert( int key, int value )

   create newNode, assign value to it,
   set newNode.next to null
   if hashTable[key] is NULL then
        hashTable[key] = newNode
   else
        Node^ currentNode = hashTable[key]
        while currentNode.next != NULL
            currentNode = currentNode.next
        end while
        currentNode.next = newNode
   end if

end insert()



int hashFunction( int value )

      calculate hashKey from value
      return hashKey

end hashFunction()



main()
      try to open input file,
      print error & exit if file not found

      initise hash table

      while file has more data...
           get hash key from hash function
           insert value into hashTable at key

      end while
      print nubmer of empty entries and collisions

end main()
```