Exercise 6 – Karp-Rabin String Search

Due by the end of your Week-9 laboratory class. (2 marks)

This exercise is to be done before or during your week 9 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. ex6.cpp, ex6.c, ex6.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 the Karp-Rabin string search algorithm. Your program should prompt for the name of an input file and then read and process the data contained in the file.

The file contains two sequences of characters on separate lines of test. The first is the target sequence, T, the second is the search sequence S. Read both strings and find all occurrences of sequence S in sequence T using the Karp-Rabin algorithm.

For each matched sequence, print the location of the first matched character in T.
For example, if the test data looked like this:

```
ACGTACGTACCAGTA
AC
```

The output should be:

```
0, 4, 8
```

Notes:
1. The sequence T is no longer than 5000 characters
2. The sequence S is no longer than 10 characters
3. The alphabet used in both sequences consists of the letters, A, C, G and T; the DNA base sequences.
4. Choose an appropriate modulus, *m*, for the hash function.
5. Try to make your hash computation as efficient as possible.

When you are finished, test your program using the provided text file named “ex6.txt” and show your code and the output to your lab tutor to receive your mark.

$ submit –u *login* –c CSCI203 –a ex6 *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

```
define the number of chars in the input alphabet d (e.g. 256)
define a suitable prime number q (e.g. 101)

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

      read the text T
      read the search pattern S and close the file
      // make sure you can access the chars with an index (e.g. T[i])

      get the length of T and S => n, m

      //calculate the hash factor
      h = pow(d, m-1)

      hash_s = hash(S,m)
      hash_t = hash(T,m])

      for (i = 0; i < n-m; i++)

            if hash_s == hash_t then
                  compare S and the substring of T to confirm
                  if match print("String found at location: " i)
            end if

            // cal next rolling hash key
            hash_t = roll( hash_t, t[i], t[i+m], h )

      end for

end main


// Rolling hash fn: Calculates hash value for next substring
int roll (int ht, int c1, int c2, int h)

      // Remove leading char and add trailing char to hash value
      ht = ( d * (ht - c1*h) + c2 )% q;
      if ht is neg add q
      return ht

end roll()


// return d exp e mod q
int pow(int d,int e)
    int p=1;
    for (int i = 0; i < e; i++)
        p = (p*d)%q;
      return p;
end pow()


// returns rolling hash hey of str
int hash( str, int m )
    int h=0;
    for (int i = 0; i < m; i++)
        h = ((d*h)+ str[i]) % q;
    return h;
end hash()
```