P5-Hash Function Implementation

Complete the MIPs code in the attached hash.s so that it implements the function given below. Note that in the example attached as a PDF which walks through the hashing for loop, the string "HAT" is hashed using M = 101 to get a hash value of 86. You can think of the string "HAT" as being expressed in base 31, so that the calculation is 'H' * 31^2 + 'A' * 31^1 + 'T' * 31^0 = 71,291. (Note: In ASCII 'A' = 65, 'B' = 66, etc.). 71,291 % 101 = 86.

Also, note that if 31 in the expression above is replaced with x, we have the form $Hx^2 + Ax^1 + T$, a polynomial. The function below evaluates the polynomial, but uses Horner's Method to reduce the number of multiplications. Thus, a polynomial such as $67x^2 + 65\ x^1 + 66$, could be represented as (x(x(67)+65)+66. This looks a lot like the code segment: **(x*h + v[i]),** where each character in v[i] is part of a string; HAT = 67,65,66.

The function to implement is shown below. For this project, use M = 104729. The hash value of "joe" should be: 679

```
int hash(char *v, int M) {
  int h=0, x =31;
  while(*v != '\0') {
    h = (x*h + *v) % M;
  v++;
  }
  return h;
}
```

Write MIPS code that uses a MIPS function (jal/jr) that will:

1) Prompt the user to input a file name
2) Read the file into memory
3) Convert the file into an integer using the above hash function.

The program should read a file and create one single integer for the FILE itself. Note that a file containing the word "joe" should hash to 679. If your program has not considered the fact that files contain invisible characters for carriage return (0x0d) and

line feed (0x0a), your hash values may be different. Write your code such that it **ignores** these characters in calculating the hash value of the file.

Expected hash values and associated files can be found below:

The three files are:

joe.txt

angie.txt

notlife.txt

For example, angie.txt appears on 2 lines as:

Angie and I

went walking

This file hashes (from C code sample) to 37867. Same as the single string "Angie and I went walking"

```
//HASHEX.C:

#include<stdio.h>
#include<stdlib.h>

#define     tblSize     104729L

unsigned int hash(char *v,  unsigned int M);

int main(int argc, char* argv[]) {
  unsigned int i=0;
  char* str[] =
  {"joe",
   "Angie and I went walking",
   "this is not life",
   "programmers are so..., -how do you say...",
   "I can't wait for the dr. who xmas special!"
  };

  for(i=0; i < 5; i++)
      printf("The hash value of %s is: %d ¥n", str[i],
hash(str[i],tblSize) );
  return 0;
}

unsigned int hash(char *v, unsigned int M)
{   unsigned int h, a = 31;
    for (h = 0; *v != '¥0'; v++)
      h = (a*h + *v) % M;
    return h;
}
/*
The hash value of joe is: 679
The hash value of Angie and I went walking is: 37867
The hash value of this is not life is: 93949
The hash value of programmers are so..., -how do you say... is: 52658
The hash value of I can't wait for the dr. who xmas special! is: 60744
```

```
*/
```

stu:

Proj 4: Hash Function Implementation

==========

REQUIREMENTS:

1. prompts user and reads in a file to memory (3pts)
2. writes result to console (1 pts)
3. correctly does computation (3 pts)
4. sufficient comments (1 pts)
5. effective use of functions/modularization (2 pts)

points given for above:

---

1. reads input: /3
2. writes output: /1
3. correct output: /3
4. comments: /1
6. modularization (jal/j ra): /2

total: x/10

"joe": 679

"Angie and I went walking": 37867

"this is not life": 93949