COS126/128: General Computer Science	Spring 1996
Problem Set # 3	
Handed out:Feb. 20, 1996	*Due: Feb. 28, 1996*

In this assignment you will translate C programs into programs for the TOY machine. This will serve two purposes: (i) It gives you a taste for how the compiler `lcc` translates your C programs into the SPARCstation's machine language. We hope this will deepen your understanding of C. (ii) Some day, faced with a crucial computational task that needs to be run as fast as possible, you may need to program in machine language. (Actually, you would use assembly language, which gives mnemonic shorthand for opcodes and distinguishes registers from constants so that it is easier for a human to deal with.) Despite many improvements in compiler technology, programs for low-level tasks written in machine language still tend to outpace the same programs written in higher-level languages.

Your TOY programs will be graded according to whether or not they are correct. You may ignore efficiency and elegance, but not good structure. You can test TOY programs by running them using the Toy machine simulator (see separate handout). For instance, to test your programs in part 1, you might wish to change the initial values of `p`, and `n`.

Part 1, 20% Translate the following C program into a corresponding TOY program.

```
#include <stdio.h>
void main()
{
  int p=11, n=55;
  if ( n%p ==  0 )
      {  printf("%d\n", p); }
  printf("%d\n", 0);
}
```

Note that the % ("modulo") operator is not part of the TOY machine's instruction set. You will need to simulate it using the existing instructions. Also note that the Toy system call for printing appends a carriage return.

Part 2, 50%: Translate the following C program into a corresponding TOY program. For full credit, you must use the JUMP-AND-LINK and JUMP-INDIRECT instructions for calls to the the function `modulo()`.

```
#include <stdio.h>
int modulo(int x, int y)
{ return x%y;}
void main()
{
  int i, n=97;  /*n is meant to be > 2 */
```

```
  for(i=2; i <= n/2; i = i+1)
      if (modulo(n,i)==0) break;
  if (modulo(n,i)==0) printf("%d\n", i); /* Print the divisor of n*/
  else printf("%d\n", 1);
}
```

Part 3, 30%: Now translate the following program into TOY language.

```
#include <stdio.h>
#define N 100
void main()
{
  int i, a[N];
  for(i=0; i < N; i=i+1)
    a[i]= i*i;
}
```

Submission: Put the three programs that you write in three separate files: `remainder.toy` with your program for part I, `prime.toy` with your toy program for part II and `array.toy` with your toy program for part III.

To electronically submit the three programs, type

`/u/cs126/bin/submit 3 remainder.toy prime.toy array.toy`

Deadline for electronic submission is 10:00pm on Wednesday, February 28, and for hardcopy submission is the lecture on Thursday, February 29.

Extra Credit:

The Toy computer could use more registers. In fact, there is room in the instruction for 16 registers – there is one unused bit in each register field. You will find the C program for the Toy simulator in /u/cs126/toy/simulator.c. Copy that program to your own directory and modify the C so that Toy has 16 registers. To help you, the relevant lines of C have been marked with a comment. Compile the new simulator and test it on a Toy program that uses more than 8 registers.