CMPS 351: Fall 2012 Homework #8
Date assigned: Thursday, September 27, 2012 Due: Thursday, October 4, 2012

The following questions ask you to write MIPS assembly programs to accomplish a simple task. Your programs should perform the requested task and cleanly exit. Your programs should include appropriate comments indicating what the code should be doing and which registers are being used. Please include your name and CLID in the program headers and include your CLID in the file names. Your programs should be turned in through Moodle before class starts on the due date. These are independent assignments meant to be solved individually, not in groups.

1. Write a MIPS assembly program that queries the user for a number and prints the factorial of that number using an iterative algorithm.
2. Write a MIPS assembly program that queries the user for a number and prints the factorial of that number using a recursive algorithm.
3. Write a MIPS assembly program that queries the user for a set of numbers and prints the sorted list using a recursive merge sort algorithm.
4. Write a MIPS assembly program to generate random integers using the Mersenne Twister algorithm.

Sample output:
Program #1:
Enter an integer: 5
5! = 120

Program #2:
Enter an integer: 5
5! = 120

Program #3:
Enter an integer: 2
Enter an integer (or 0 to stop): 4
Enter an integer (or 0 to stop): 3
Enter an integer (or 0 to stop): 7
Enter an integer (or 0 to stop): 5
Enter an integer (or 0 to stop): 13
Enter an integer (or 0 to stop): 2
Enter an integer (or 0 to stop): 0

The sorted list is {2, 2, 3, 4, 7, 5, 13}.

Program #4:
How many random numbers would you like to generate?
3

799760409
22388334
2083618663

C code for the Mersenne Twister PRNG algorithm:
(This algorithm can also be found in the MT paper on Moodle.)

```
/* A C-program for MT1999937:  Integer number version                   */
/* genrand() generates one pseudorandom integer number (int) which is  */
/* uniformly distributed on [0, maxint]-interval, for each call.       */
/* sgenrand(seed) set initial values to the working area of 624 words. */
/* Before genrand(), sgenrand(seed) must be called once.  (seed is any */
/* 32-bit integer except for 0).  Real generator is obtained by        */
/* modifying two lines.                                                */
/*   Coded by Takuji Nishimura, considering the suggestions by Topher  */
/* Cooper and Marc Rieffel in Jul-Aug. 1997.  Comments should be       */
/* addressed to:  matumoto@math.keio.ac.jp                             */

#include<stdio.h>

/* Period parameters */
#define N 624
#define M 397
#define MATRIX_A 0x9908b0df   /* constant vector a */
#define UPPER_MASK 0x80000000 /* most significant w-r bits */
#define LOWER_MASK 0x7fffffff /* least significant r bits */
/* Tempering parameters */
#define TEMPERING_MASK_B 0x9d2c5680
#define TEMPERING_MASK_C 0xefc60000
#define TEMPERING_SHIFT_U(y)  (y >> 11)
#define TEMPERING_SHIFT_S(y)  (y << 7)
#define TEMPERING_SHIFT_T(y)  (y << 15)
#define TEMPERING_SHIFT_L(y)  (y >> 18)

static unsigned long mt[N];  /* the array for the state vector */
static int mti=N+1;  /* mti=N+1 means mt[N] is not initialized */

/*  initializing the array with a NONZERO seed */
void
sgenrand(seed(
      unsigned long seed;
```

```
{
      /* setting initial seeds to mt[N] using the generator Line 25 of */
      /* Table 1 in [KNUTH 1981, The Art of Computer Programming       */
      /*    Vol. 2 (2nd Ed.), pp102]                                   */
      mt[0] = seed & 0xffffffff;
      for (mti=1; mti<N; mti++)
            mt[mti] = (69069 * mt[mti-1]) & 0xffffffff;
}

/* double */ /* for generating reals */
unsigned long /* for integer generation */
genrand()
{
      unsigned long y;
      static unsigned long mag01[2]={0x0, MATRIX_A};
      /* mag01[x] = x * MATRIX_A_ for x=0,1 */

      if (mti >= N) { /* generate N words at one time */
            int kk;

            if (mti == N+1)       /* if sgenrand() has not been called,*/
                  sgenrand(4357);  /* a default initial seed is used   */

            for (kk=0; kk<N-M; kk++) {
                  y = (mt[kk]&UPPER_MASK) | (mt[kk+1]&LOWER_MASK);
                  mt[kk] = mt[kk+M] ^ (y >> 1) ^ mag01[y & 0x1];
            }
            for (; kk<N-1; kk++) {
                  y = (mt[kk]&UPPER_MASK) | (mt[kk+1]&LOWER_MASK);
                  mt[kk] = mt[kk+(M-N)] ^ (y >> 1) ^ mag01[y & 0x1];
            }
            y = (mt[N-1]&UPPER_MASK) | (mt[0]&LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >> 1) ^ mag01[y & 0x1];

            mti = 0;
      }

      y = mt[mti++];
      y ^= TEMPERING_SHIFT_U(y);
      y ^= TEMPERING_SHIFT_S(y) & TEMPERING_MASK_B;
      y ^= TEMPERING_SHIFT_T(y) & TEMPERING_MASK_C;
      y ^= TEMPERING_SHIFT_L(y);

      /* return ( (double)y / (unsigned long)0xffffffff ); */ /* reals */
      return y; /* for integer generation */
}
```

CMPS 351: Fall 2012 Homework #8
Date assigned: Thursday, September 27, 2012 Due: Thursday, October 4, 2012

**Objectives**:

1. To introduce and practice with loops in the MIPS assembly language.
2. To introduce and practice with recursive procedures in the MIPS assembly language.
3. To introduce and practice with arrays and memory management in the MIPS assembly language.
4. To introduce and practice with MIPS logical operations.

**Point Values**:

1. 25pts
2. 25pts
3. 100pts
4. 50pts

Total. 200pts