Burt Rosenberg

Merge.c

Out: July 3, 2002

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

#define ARRAY_SIZE 32
#define MULTIPLIER 15
#define MODULUS    101
#define NUM_PER_COL 8
/*
   merge sort sample program
   burt rosenberg
   math 220, oct 1995
*/


void merge_arrays( int A[], int start, int num, int T[] )
{
   /* merge two lists into one. The lists are in the array
      A from start ... start+(num/2)-1
      and start+num ... start+2(num/2)-1,
      T is a temporary array, of the same dimension of A
   */

   int l_a ;
   int l_b ;
   int i = 0 ;

   l_a = start ;
   assert( num%2==0 ) ;
   l_b = start+ num/2 ;

   while ( l_a < start+num/2 && l_b < start+num )
   {
      if ( A[l_a]<A[l_b] )
      {
         /* move A[l_a] to T[i] */
         T[i] = A[l_a] ;
         l_a = l_a + 1 ;
      }
      else
      {
```

```
         /* move A[l_b] to T[i] */
         T[i] = A[l_b] ;
         l_b = l_b + 1 ;
      }
      i = i + 1 ;
   }
   /* the goal is to assert that l_a == start+num/2 and l_b == start+num
      make this true */
   while ( l_a < start+num/2 )
   {
      T[i] = A[l_a] ;
      l_a = l_a + 1 ;
      i = i + 1 ;
   }
   while ( l_b < start+num )
   {
      T[i] = A[l_b] ;
      l_b = l_b + 1 ;
      i = i + 1 ;
   }
   assert( l_a==start+num/2 && l_b==start+num ) ;

   /* copy T into A */
   for ( i=0; i<num; i++ )
   {
      A[start+i] = T[i] ;
   }
}

void print_array( int A[], int num )
{
   int i ;
   for ( i=0; i<num; i++ )
   {
      printf("%7d ",A[i]) ;
      if ( (i+1)%NUM_PER_COL==0 )
      {
         printf("\n") ;
      }
   }
   printf("\n") ;
}

void merge_sort_aux( int A[], int start, int num, int T[] )
```

```
{
   /* called by merge sort, will sort A[start]...A[start+num-1]
      by recursive calls to first and second half of the
      interval start...start+num-1, then merging.
   */

   /* ASSERT num = 2^i for some i */
   if ( num==1 ) return ;
   assert(num>1) ;

   /* sort first half */
   merge_sort_aux( A, start, num/2, T ) ;
   /* sort second half */
   merge_sort_aux( A, start+num/2, num/2, T ) ;

   merge_arrays( A, start, num, T ) ;

   return ;
}

void merge_sort( int A[], int num )
{
   /* set up temporary storage and initialize recursion
      variables, then call the auxillary function.
   */
   int * T ;
   T = (int *)malloc(num*sizeof(int)) ;
   merge_sort_aux( A, 0, num, T ) ;
   free(T) ;
}

int main( int argc, char * argv[] )
{
   int A[ARRAY_SIZE] ;
   int i ;
   for ( i=0; i<ARRAY_SIZE; i++ )
   {
      A[i] = (i+1)*MULTIPLIER % MODULUS ;
   }
   printf("Array before merge sort.\n") ;
   print_array( A, ARRAY_SIZE ) ;
   merge_sort( A, ARRAY_SIZE ) ;
   printf("Array after merge sort.\n") ;
   print_array( A, ARRAY_SIZE ) ;
```

```
}

passaic> cc merge.c
passaic> a.out
Array before merge sort.
     15      30      45      60      75      90       4      19
     34      49      64      79      94       8      23      38
     53      68      83      98      12      27      42      57
     72      87       1      16      31      46      61      76

Array after merge sort.
      1       4       8      12      15      16      19      23
     27      30      31      34      38      42      45      46
     49      53      57      60      61      64      68      72
     75      76      79      83      87      90      94      98
```