CS13002 Programming and Data Structures, Spring 2006

Lab test 3

Total points: 30 | April 4, 2006 | Time: 13:30–16:00

For students with odd PC numbers

In this exercise, you are asked to write a C program that generates a linked list of integers and subsequently splits the list in two sublists, the first containing the odd integers and the second containing the even integers in the list. Do not create fresh lists by allocating memory, but adjust the pointers of the original list in order to generate the two output lists. Print the input list (before splitting) and both the output lists (after splitting).

For your convenience a structure of the program is provided in the next page. Fill out the details of the split and print functions in accordance with the prototypes mentioned. Your input list should consist of 100 elements between 0 and 999 with repetitions allowed. Maintain a *dummy node* at the beginning of each list.

Sample output

The following output corresponds to an input list of size 50.

```
Original list :  306 394   6 566 255 467 870  97 944 208 577 412 443 43169 407
 534 837 606 646 330  54  56 920 245  56 712 958 431   0 411 737 394 770 655  1
 589 525 450 533  85 379 297 881 914 466 288 800 655 895
Number of elements = 50

Sublist 1     :  255 467  97 577 443 169 407 837 245 431 411 737 655   1 589525
 533  85 379 297 881 655 895
Number of elements = 23

Sublist 2     :  306 394   6 566 870 944 208 412 534 534 606 646 330  54  56 29
  56 712 958   0 394 770 450 914 466 288 800
Number of elements = 27
```

Notice that your print-out can display a maximum of 80 characters per line. Insert appropriate line breaks. One strategy is to use four characters (`%4d`) for printing each integer and insert a line break after 20 integers are printed.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define LIST_SIZE 100

typedef struct _node {
   int element;
   struct _node *next;
} node;

/* Create a list of n integers with a dummy node*/
node *createList ( int n )
{
   int i;
   node *L, *p;

   p = L = (node *)malloc(sizeof(node));
   for (i=0; i<n; ++i) {
      p -> next = (node *)malloc(sizeof(node));
      p = p -> next;
      p -> element = rand() % 1000;
   }
   p -> next = NULL;
   return L;
}

/* Print the elements of a list from beginning to end*/
void printList ( node *L ) ;

/* Split a list L in two sublists L1,L2 as specified in the text above */
void splitList ( node *L , node *L1 , node *L2 ) ;

int main ()
{
   node *L, *L1, *L2;

   /* Seed the random number generator by system time*/
   srand((unsigned int)time(NULL));

   /* Create and print the input list L*/
   L = createList(LIST_SIZE);
   printf("\nOriginal list : "); printList(L);

   /* Create the dummy nodes for the output lists L1,L2*/
   L1 = (node *)malloc(sizeof(node));
   L2 = (node *)malloc(sizeof(node));

   /* Split the input list L and print the output lists L1,L2*/
   splitList(L,L1,L2);
   printf("\nSublist 1     : "); printList(L1);
   printf("\nSublist 2     : "); printList(L2);
}
```