Burt Rosenberg

Problem Set 2

Out: 5 September, 1995
Due: 12 September, 1995

Reading Assignment

Chapters 6, 7, 8, 11, 12 and 16 in the cow book.

Programming Assignment

1. Write a function that duplicates a list. It need not maintain the orignal order of elements on the list.
2. Improve upon (1) to duplicate a list while maintaining the original order of elements.
3. Write a function that finds, prints and deletes the minimum integer on a list.
4. Use (3) to print a list in ascending order, deleting the list as you go.

Example Programs

```
% cat test3.c
#include<stdio.h>   /* include standard IO libraries */
#include<stdlib.h>  /* ANSI C, needed for malloc */

/* test3.c
   Burt Rosenberg
   Mth 220, Fall 1995
*/

struct MyList {          /* MyList is the structure tag */
   int the_number ;      /* an integer field, called the_number*/
   struct MyList *next ; /* a pointer to another struct MyList */
} ;

struct MyList *anchor ;  /* make a place for a pointer to a struct */
struct MyList *l_tmp ;   /* a temporary pointer to structure,
                            because we don't yet know about making
                            variables local to functions */

/* functions to be defined later */
struct MyList *push_MyList( int i, struct MyList *l ) ;
void print_MyList( struct MyList *l ) ;
```

```
main() {
  anchor = NULL ;        /* initialize to an empty list */
  /* add 11, 7 and 3 to the front of the list */
  anchor = push_MyList( 11, anchor ) ;
  anchor = push_MyList( 7, anchor ) ;
  anchor = push_MyList( 3, anchor ) ;
  /* print the list */
  print_MyList( anchor ) ;
}

/* push_MyList:
      takes an integer and a list and adds a new
      element, containing the integer, to the head
      of the list. Returns the new head of the list.
*/
struct MyList *push_MyList( int i, struct MyList *l ) {
  /* first, create the structure */
  l_tmp = (struct MyList *) malloc( sizeof(struct MyList) ) ;
  /* sizeof tells malloc how large, (struct MyList *) casts
     the type */
  (*l_tmp).the_number = i ;  /* fill in field with number */
  /* this can also be written l_tmp->the_number = i */
  (*l_tmp).next = l ;        /* fill in field with list pointer */
  /* this can also be written l_tmp->next = l */
  return( l_tmp ) ;          /* return the head of new list */
}

/* print_MyList:
      prints the MyList
*/
void print_MyList( struct MyList *l ) {
   while( l!=NULL ) {                  /* while not done ... */
      printf("%d ", (*l).the_number ) ; /* print the_number field */
      l = (*l).next ;                  /* then move to next struct */
      /* NOTE: the caller's argument is not changed, l is local */
   }
   printf("\n") ;
}
% cc test3.c
% a.out
3 7 11
%
```