CS 1723, Data Structures, Fall 1998
Selected Answers to Final Examination

PROBLEM 1: Strlen

```
vip% cat final1.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int strlen1(char *);  /* strdup1 and strlen1 to be */
void main(void)
{
   char s[] = "Rats&roaches";
   printf("%i\n", strlen1(s));
}

int strlen1(char *s)
{
   int len = 0;
   while (*(s++)) /* or while(*s++ != '\0') */
      len++;
   return len;
}

vip% cc -o final1 final1.c
vip% final1
12
```

PROBLEM 3: Linked list of characters

```
runner% cat final_5.c
#include <stdio.h>
#include <stdlib.h>

struct lnode {
        char data;
        struct lnode *next;
};

int listlength(struct lnode *list);
struct lnode *makelist(char *s);
void printlist(struct lnode *list);

void main(void)
{
        struct lnode *list;
        char t[] = "Why worry?";
        list = makelist(t);
        printlist(list);
        printf("%d\n", listlength(list));
}

int listlength(struct lnode *list)
{
        int len = 0;
        while (list != NULL) {
                list = list -> next;
                len++;
        }
        return len;
}

void printlist(struct lnode *list)
{
        while (list != NULL) {
                printf("%c", list -> data);
                list = list -> next;
        }
        printf("\n");
}

struct lnode *makelist(char *s)
{
        struct lnode *oldlist = NULL;
        struct lnode *newlist;
        while (*s != '\0') {
                newlist = (struct lnode *)
                          malloc(sizeof(struct lnode));
                newlist -> data = *s;
                newlist -> next = oldlist;
                oldlist = newlist;
                s++;
        }
        return newlist;
}
runner% cc -o final_5 final_5.c
runner% final_5
?yrrow yhW
10
```

PROBLEM 4: Command line arguments.

```
runnner% cat final3.c
#include <stdio.h>
void printit(char *argv[]);
void main(int argc, char *argv[])
{
   char *myargv[] = {"mycc", "-o", "prog", "prog.c",
      NULL};
   printit(argv);
   printit(myargv);
}

void printit(char *argv[])
{
   int line = 0;
   char **p = argv;
   printf("Array of Strings:\n");
   while (*p) {  /* or while(*p != NULL) */
      printf("  Line %d:\"%s\"\n", line, *(p++));
      line++;
   }
}
runner% cc -o final3 final3.c
runner% final3 -o prog prog.c  (Extra args not used)
Array of Strings:
  Line 0:"final3"
  Line 1:"-o"
  Line 2:"prog"
  Line 3:"prog.c"
Array of Strings:
  Line 0:"mycc"
  Line 1:"-o"
  Line 2:"prog"
  Line 3:"prog.c"
```

PROBLEM 5:

```
runner% cat exam2_3.c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
```

```
struct tnode {
   char ch;
   struct tnode *left;
   struct tnode *right;
};
void          addtree(struct tnode *, char );
struct tnode *newnode(char );
void          treeprint(struct tnode *);
struct tnode *lookup(struct tnode *, char );

void main(void)
{
   struct tnode *root;
   struct tnode *p;
   char c;
   root = NULL;
   while ((c = getchar()) != '\n') {
      if (root == NULL) root = newnode(c);
      else addtree(root, c);
   }
   treeprint(root);
   printf("\n");
   p = lookup(root, 'x');
   if (p == NULL) printf("Not found\n");
   else printf("Here it is: %c\n", p -> ch);
}

/* addtree: add a node, non-recursive */
void addtree(struct tnode *p, char c)
{
   for (;;) {
      if ((p -> ch) == c) return; /* ignore */
      else if ((p -> ch) > c) { /* go left */
         if ((p -> left) == NULL) {
            p -> left = newnode(c);
            return;
         }
         else p = p -> left;
      }
      else if ((p -> ch) < c) { /* go right */
         if ((p -> right) == NULL) {
            p -> right = newnode(c);
            return;
         }
         else p = p -> right;
      }
   }
}

/* lookup: find the node of a char c */
struct tnode *lookup(struct tnode *p, char c)
{
   for (;;) {
      if ((p -> ch) == c) return p;
      if ((p -> ch) > c) { /* go left */
         if ((p -> left) == NULL) {
            return NULL;
         }
         p = p -> left;
      }
      else if ((p -> ch) < c) { /* go right */
         if ((p -> right) == NULL) {
            return NULL;
         }
         p = p -> right;
      }
   }
}

/* newnode: fix up a new node */
struct tnode *newnode(char c)
{
   struct tnode *p = (struct tnode *)
        malloc(sizeof(struct tnode));
   p -> ch = c;
   p -> left = p -> right = NULL;
   return p;
}

/* treeprint: recursive inorder print of tree p */
void treeprint(struct tnode *p)
{
   if (p != NULL) {
      treeprint(p -> left);
      printf("%c", p -> ch);
      treeprint(p -> right);
   }
}

runner% cc -o exam2_3 exam2_3.c
runner% exam2_3
The quick brown fox jumps over the lazy dog.
 .Tabcdefghijklmnopqrstuvwxyz
Here it is: x
runner% exam2_3
The quick brown pig jump up in the air.
 .Tabceghijkmnopqrtuw
Not found
runner%
```

```
PROBLEM 6:  Recursive binary search function:
  /* from main function */
   index = (int)(M*drand48());
   x = a[index];
   result = bin_search(a, 0, M-1, x);
   printf("Search for: %d, found at %d, same as %d\n",
      x, result, index);
  ...
/* bin_serach: search for x in a from p to r */
int bin_search(int a[], int p, int r, int x)
{
   int q;
   if (p > r) return -1;
   q = (p + r)/2;
   if (a[q] == x) return q;
   if (x < a[q])
      return bin_search(a, p, q-1, x);
   /* else if (x > a[q]) */
   return bin_search(a, q+1, r, x);
}
runner% final_3
Array sorted, 100000 items
Search for: 24311336, found at 24295, same as 24295
```