```
runner% cat structs.c
/* structs.c: study the struct in C */
/* Written 4 Mar 1997 by NR Wagner */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct a_tag {
   char c;
   int i;
   char *w;
};

void print_a(int test_num, struct a_tag a);
void print_p(int test_num, struct a_tag *p);
struct a_tag  return_struct(void);
struct a_tag *return_ptr(void);
struct a_tag *return_bad(void);

int main()
{
   struct a_tag a1, a2, a3; /* structs */
   struct a_tag *p1 = &a1;  /* p1 points to a1 */
   struct a_tag *p3, *p4;   /* pointers */
   /* initialize a1 */
   a1.c = 'X'; a1.i = 99;
   a1.w = (char *) malloc(20);
   strcpy(a1.w, "Now is the time");
   /* printing related to a1, using p1 also */
   /* shows that print_a copies struct, while */
   /* print_p copies a pointer to the struct */
   print_a(1, a1);
   print_p(2, p1);
   print_a(3, a1);
   print_p(4, p1);
   /* = copies struct on right into struct on left */
   a2 = a1;
   print_a(5, a2);
   /* change a1 and a2 gets changed also! */
   strcpy(a1.w, "Quick brown foxes");
   print_a(6, a2);
   /* try out returns, first returning a struct */
   a3 = return_struct();
   print_a(7, a3);
   /* then try returning a pointer to a struct */
   p3 = return_ptr();
   print_p(8, p3);
   /* finally return a pointer to local auto storage */
   p4 = return_bad();
   print_p(9, p4);
   return 0;
}

/* print_a: print a struct passed by value (copied) */
void print_a(int test_num, struct a_tag a)
{
   printf("Test %i, ", test_num);
   printf("Passing struct,  fields: c:%c, i:%3i, w:\"%s\"\n",
      a.c, a.i, a.w);
   /* increment below has no effect back in main */
   a.i++;  /* same as (a.i)++ */
}
```

```
/* print_a: print a struct where pointer is passed */
void print_p(int test_num, struct a_tag *p)
{
   printf("Test %i, ", test_num);
   printf("Passing pointer, fields: c:%c, i:%3i, w:\"%s\"\n",
      (*p).c, p -> i, p -> w);  /* (*p).c same as p -> c */
   /* increment below changes struct back in main */
   p->i++; /* same as (p->i)++ */
}

/* return_struct: return a struct by value (copied) */
struct a_tag return_struct(void)
{
   struct a_tag a;
   a.c = 'Y';
   a.i = 88;
   a.w = "My dog has fleas";
   /* can return local auto storage, because it is copied */
   return a;
}

/* return_ptr: return a pointer to a struct */
struct a_tag *return_ptr(void)
{
   struct a_tag *p;
   p = (struct a_tag *) malloc(sizeof(struct a_tag));
   p -> c = 'Z';
   (*p).i = 77; /* same as p -> i = 77 */
   p -> w = "Yours does too";
   return p;
}

/* return_bad: return local auto storage through */
/*   a pointer to a struct--a serious error! */
struct a_tag *return_bad(void)
{
   struct a_tag *p;
   struct a_tag a;
   p = &a;
   p -> c = 'U';
   (*p).i = 666; /* same as p -> i = 66 */
   p -> w = "Should screw up";
   return p;
}
runner% lint -m -u structs.c

function returns value which is always ignored
    printf              strcpy
runner% cc -o structs structs.c
runner% structs
Test 1, Passing struct,  fields: c:X, i: 99, w:"Now is the time"
Test 2, Passing pointer, fields: c:X, i: 99, w:"Now is the time"
Test 3, Passing struct,  fields: c:X, i:100, w:"Now is the time"
Test 4, Passing pointer, fields: c:X, i:100, w:"Now is the time"
Test 5, Passing struct,  fields: c:X, i:101, w:"Now is the time"
Test 6, Passing struct,  fields: c:X, i:101, w:"Quick brown foxes"
Test 7, Passing struct,  fields: c:Y, i: 88, w:"My dog has fleas"
Test 8, Passing pointer, fields: c:Z, i: 77, w:"Yours does too"
Test 9, Passing pointer, fields: c:õ, i: 85, w:"õ þp"
runner% CC -o structscc structs.c
runner% structscc  (same output)
runner% gcc -o structsgcc structs.c
runner% structsgcc
. . .
Test 9, Passing pointer, fields: c:, i:68608, w:""
```