CSCI 254 -- C Programming Language

Final Exam

6 December 1990, 3:05-5:30 pm

Problem 1: (10 points)

In this question you will consider the effect of reading a single input line by several different **scanf** statements. The following line (which does *not* begin with a blank) is the input line:

3.14 1066 how now

The variables that will be used in the **scanf** statements are declared as:

```
float f;
int i, j;
char c, s[5];
```

What are the values assigned to these variables by executing the following five **scanf** statements with the common line of input?

```
(a)  scanf("%f%d%s", &f, &i, s);
(b)  scanf("%3f%d %d%s", &f, &i, &j, s);
(c)  scanf("%d%c%o%f", &i, &c, &j, &f);
(d)  scanf("3.%x %2d%fhow n%s", &i, &j, &f, s);
(e)  scanf("%f %d brown cow %s", &f, &i, s);
```

Problem 2: (10 points)

Given the following variable declarations:

```
int i, j;
float f, g;
```

what is the effect of executing, *in order*, the following five statements?

```
(a)  f = (g=(j=5) + (i=1)) ;
(b)  i = (f > g) + j ;
(c)  f = i/j + g/4 ;
(d)  i = 1<<j + 1 | j ;
(e)  i = (i>0 && j==3) + 7 ;
```

Problem 3: (20 points)

Write a function which takes as its single argument an array of 100 integers and returns the second largest integer in the array.

Problem 4: (15 points)

In a widely-used electronic mail message standard called RFC822, an electronic mail address is represented as:

PERSON-NAME <COMPUTER-MAILBOX>

that is, a person's real name followed by the name of their computer mailbox enclosed in angle brackets.

Your task is to write a function called **GetMailbox** which takes as its single argument a pointer to a string of characters in RFC822 form and returns a pointer to a string containing the electronic mail box. For example, given the argument:

Dean Brock <brock@cs.unca.edu>

your function should return

brock@cs.unca.edu

If you use the string functions **strchr** and **strcpy**, you can do this problem with about eight lines of C. However, if you do the problem perfectly (that is, without permanently changing the input string), you'll have to use the **malloc** function and about ten lines of C.

Problem 5: (10 points)

Write a few lines of C code that opens a file called **"open.me"** and reads the first character in the file.

Problem 6: (15 points)

Start with the C structure definition:

```
struct intlist {
   int value;
   struct intlist *next;
  }
```

Now look at the following recursive C function.

```
struct intlist *OddList(int n)
 {
  struct intlist *t;
  if (n < 1)
    return((struct intlist *) NULL) ;
  else
   {
    t = (struct intlist *)malloc(sizeof(struct intlist)) ;
    t->value = n;
    t->next = OddList(n-2);
   }
 }
```

Draw the data structure returned by the call **OddList(5)**.

Problem 7: (20 points)

Turbo C provides some nonstandard time and date functions that similar to those provided by MS/DOS. The structure used by the data functions is defined as:

```
struct date {
     int da_year;         /* year */
     int da_day;          /* day of month */
     int da_mon;          /* month */
  }
```

Turbo C has a function getdate with prototype:

```
     void getdate(struct date *d)
```

which "fills the date structure pointed to by **d** with the DOS form of the current system date" [Schildt, *Turbo C: The Complete Reference*].

Write a function called **celebrate** using the following header:

```
     int celebrate(struct date *DayOfBirth)
```

Your function should return 1 when called on the birth date of someone born on the day represented its parameter **DayOfBirth** and 0 otherwise.