csci-e113 20 November 2000

Midterm Exam

Your Name:

Instructions: You have two hours for this exam. You may use any notes or texts you like. Please put your answers in the spaces provided on this test sheet. The point values of the questions are listed in the table below. Watch the time, write clearly, good luck.

#	*points*	*score*	*totals*
1	4		
24			
34			
44			
54			
64			
75			
85			
95			
10	9		
pics	6		
&t	3		
11A	8		
11B	8		
11B 8			
12A	6		
12B	6		
13	7		

Problems 1-6: Compare and Contrast. For each of the following pairs of six fragments, describe the difference in value, effect, and/or utility. Please be concise and stay in the space provided.

1. A.
```
break;
```
B.
```
continue;
```

2. A.
```
x += x;
```
B.
```
x++;
```

3.
```
/* declarations */
```
A.
```
int *f(char x);
```
B.
```
int f(char *x);
```

4. A.
```
p = x.next;
```
B.
```
p = x->next;
```

5. A.
```
int x[100];
```
B.
```
extern int x[100];
```

6. A.
```
char dog[] = "betsy";
```
B.
```
char *dog  = "betsy";
```

Problems 7-9: Syntax/Usage. Something is wrong with each of the following C fragments. State the problem AND correct the error.

7.

```
void print_string(char str[])
{
   char *p = str;
   int  i  = 0;
   while( str[i] != '\0' )
        putchar( *p++ );
}
```

8.

```
char *p;
p = malloc(1 + strlen(str));
p = str;
```

9.

```
while( getchar() != EOF )
    if ( getchar() == '\n' )
        num_lines++;
```

Problem 10: Arrays and Strings: Given that the output from running this function:

```
f()
{
      char month[20], *s, *t, c;
      strcpy(month, "November");
      s = month;
      t = ++s;
      c = ++*t;
      printf("month = %u,  &s = %u\n", month, &s);
}
```

is `month = 48708,  &s = 48704`

State the values of each of these expressions (use letters for char values):

a) `month[3] = _____` b) `month+3 = _____` c) `*(month+3) = _____`

Draw a diagram showing all four of these variables as rectangles. For each variable, draw in the picture of the variable the value stored in that variable at the time just after the printf() statement runs. (use letters, not ASCII codes for chars). **Also** where possible, show the address in memory of the variable.

What do you think is the value of `&t` ? Why?

Problem 11: Working with Strings

The study of DNA has been aided by computers. Genes are long sequences of a few basic molecules, each represented by a single letter. In C, you represent a sequence of letters as a string. How can you compare two strings to see how similar they are? The function `strcmp()` is not useful here, because it tells you which string comes before the other in a dictionary. We need a different sort of function for genetic research, one that compares two strings to see how many characters they have in common or at how many positions they differ.

For this problem, write two versions of the function described here:

> **int strdiff(char s[], char t[])** returns a number telling at how many places the strings *s* and *t* differ. If the strings have different lengths, the 'missing' characters in the shorter string are considered to be differences.

Part A Write a version of `strdiff()` that uses array indexing to process the strings. Write the complete function not just the body of the code.

Part B Write a version of `strdiff()` that uses pointer operations to process the strings. Write the complete function, not just the body of the code.

Part C Write a new function called `strsame(char s[],char t[])` that returns the number of positions at which two strings are the same. Assume for this function that both strings are of equal lengths. This function **must** make a call to `strdiff()`.

Problem 12: Arrays and Structs

The `strdiff()` function you wrote for problem 11 has applications outside of genetic research. You decide to build some general tools to use it. A new system stores pairs of strings in structs:

```
struct pair {
               char *s1;            /* first string  */
               char *s2;            /* second string */
               int  diffs;          /* number of differences */
               struct pair *next;   /* for linking   */
};
```

As pairs of strings come in, the system needs to store them, compute the number of characters where they differ, and then add them to a linked list. You also want a function to search the linked list for close matches. You will write two of these functions.

Part A Write a function `struct pair *newpair(char s[], char t[])` that creates one of these structs. The function takes as input two strings. It allocates space for the struct and for the two strings, installs the strings, and it records the number of differences. It then returns a pointer to this new struct.

Part B Write a function `print_similar(struct pair *first, int maxdiffs)` that traverses a linked list of these structs and prints out any pairs of strings for which the number of differences is less than or equal to *maxdiffs*. The function is passed a pointer to the first link in the list and the maximum number of differences accepted. The output is of the form

```
diffs  string1  string2    for example:    3  tunafish  monkfish
```

Problem 13: Uncounting

The `uniq -c` takes a sequence of lines as input and writes as output the number of repetitions and the text of the line. This system of writing out a count and a string is a form of compression called 'run length encoding'.

Write a program to 'uncompress' the output of `uniq -c`. Your program should read from standard input lines of the form:

```
4 spam and cheese
3 spam and egg
2 spam and toast
```

and write to standard output each line repeated as many times as specified by the count.

Your program should be a complete, ready to compile and run C program.