csci-e113 16 November 1998

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	18		
11A	10		
11B	10		
12A	5		
12B	10		
13	8		

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.  char x[100];

B.  char *x;
```

2.
```
char *p1, *p2;

A.  if ( p1 == p2 )
        f();

B.  if ( strcmp(p1,p2) == 0 )
        f();
```

3.
```
A.  return(0);

B.  exit(0);
```

4.
```
A.  x = --y ;

B.  x = y - 1;
```

5.
```
A.  fscanf(stdin,"%s", buffer);

B.  fgets(buffer, LEN, stdin);
```

6.
```
char *str, *p;
A.  p = malloc(strlen(str));

B.  p = malloc(sizeof(str));
```

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

7.

```
void copy_string(char *str1, char *str2)
{
   int i;
   for( i = 0 ; str1[i] ; i++ )
     str2[i] = str1[i];
}
```

8.

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

9.

```
while( c = getchar() != EOF )
    putc(c, stdout);
```

Problem 10: Arrays and Strings: Given the following definitions and initial assignments,

```
char *s[4] = { "fall", "winter", "summer", "spring" };
int  j = strlen( s[0] );
char *p = s[1] , *q = p + 1;
```

state the value of each of the expressions a) - d): a) `p[0] =` ___ b) `p[j] =` ___ c) `*q =` ___ d) `s[1][1] - s[3][3] =` ___
e) what is the *type* of the expression `&s[2]`? _____
f) Notice that "summer" and "spring" are in the wrong order in this array. Write a code fragment that will exchange the order of those two words.

Problem 11: Arrays and Pointers

Part A Consider the array `{ 2,3,5,10,12,15 }`. It is a list of numbers in ascending order. On the other hand,the array `{ 2,3,5,4,12,15 }` is not in ascending order because 4 is less than 5. Write a function

```
int in_order(int list[], int len )
```

that determines if the array of integers 'list' is in ascending order. The array contains 'len' elements. 'len' may be zero. The function returns 1 if the elements in the array are in ascending order, and it returns 0 if not.

Part B Write a function to add together two arrays of integers. For example, if the arrays are `{1,2,3,4,0}` and `{2,3,4,5,0}` then the result will be the array `{3,5,7,9,0}`. The function should add arrays by adding corresponding elements. The function takes three arguments: the two arrays to add and an array for the result. Each array is terminated by an element equal to zero, so there is no need to send in the length of the array. Your solution *must* use pointers and pointer operations. It may not use array indexing.

Problem 12: Memory and Structs

A computer science teacher teaches a class of `NUM` students and assigns six programming assignments and two tests during the term. He designs the following data structure to store student information:

```
struct student
{
    int   idnumber;
    char  *name ;
    int   grades[6];
    int   tests[2];
}
struct student class[NUM];
```

a) Sketch a diagram of this data structure. You do not need to draw all 25 records, one or two will do.

b) Write a function

```
print_hw_grades( struct student class[], int hw_number , int len)
```

that prints a report of the form:

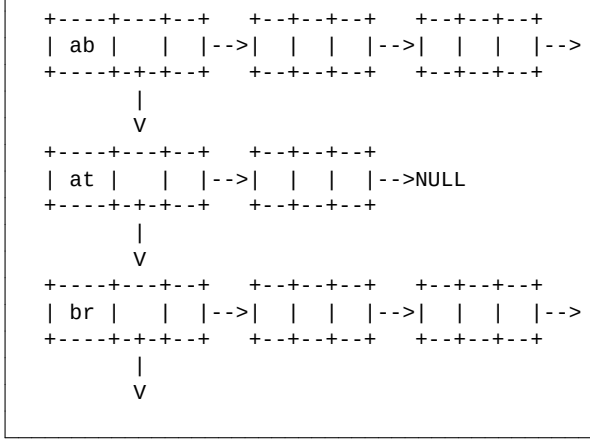
```
HW#2 Name
  92 Barbara
  88 Charlie
  90 Edith
 ...
```

The function is passed the list of students, an assignment number, and the number of students in the class. The assignment number is an integer in the range 0 to 5.

Problem 13: Linked Lists

The wlfiler5.c package used an array of 26 linked lists to store word-value pairs. This system speeds up searching by breaking the list of words into 26 separate lists. If the number of words grows large each of those 26 lists could become long and slow to search. One solution is to replace the array of linked lists with a linked list of linked lists.

Picture



Data Types

```
struct link
{
   char        *word;   /* ptr to word      */
   int         value;   /* word frequency   */
   struct link *next;   /* ptr to next word */
} ;

struct list
{
   char        *prefix; /* bucket prefix    */
   struct list *next  ; /* ptr to next list */
   struct link *words ; /* ptr to words     */
}

struct list     head ;  /* head of list of lists   */
```

This system uses a two-letter prefix to identify each list. All the words that start with ‘ab’ are in the list pointed to by the `struct list` containing the prefix "ab". Thus, the words ‘about’, ‘able’, ‘abound’ are all on the first list. All words starting with "at" are in the second list. And so on.

Write a function to print out all the words in this ‘table’ that have a frequency of 10 or greater.