

NAME _____

I. (64 points.) ****SHOW ALL WORK FOR PARTIAL CREDIT****

(1) (16 points) What is printed out by the following code fragment? Start by showing how you would parenthesize these expressions to AGREE with the precedence table on page 53 of K&R, then show your calculations of intermediate results (use BINARY when possible).

```
unsigned char c;
```

```
  . . .  
c = '\123';
```

```
c = c % 0X40 ^ '\x2D' << 1;
```

```
printf("Value one is %o, Value two is %d, Value three is %x\n", 2*c, 3*c, c);
```

Answer:

(2) (16 points) Consider the array months[] below:

```
char months[ ] = "Jan\0Feb\0Mar\0Apr\0May\0Jun\0Jul\0Aug\0Sep\0Oct\0Nov\0Dec\0";
```

(a) (4 points) Write a printf statement to print out the string "Feb" from this array.

Answer:

(b) (5 points) Write a declaration for an int array "monthdays[]" with initialization to set array values to the number of days in the months Jan (in monthdays[1]) through Dec (in monthdays[12]) of a year that is NOT a leap year; use initialization to set monthdays[0] = 0.

Answer:

(c) (7 points) Write a **SINGLE C STATEMENT (IMPORTANT!)** using an int value year (e.g., year = 1999), that modifies the monthdays[] array appropriately according to whether year represents a leap year or not.

Answer

(3) (12 points) Consider the following declarations in two different source files. First in source file main.c:

```
static int p;  
int q, m;  
  
main ( )  
{  
    int n, r;  
    static int k;  
}
```

In a different source file, func.c, we have declarations outside any function (these don't all work):

```
extern int q, m;  
extern int n, r;  
extern int k;  
extern int p;
```

Of the above variables, name those that are automatic. _____

Name those that have a permanent value (storage location not on stack). _____

Name those where the extern declarations in func.c will work. _____

(4) (20 points) Write a VERY SHORT function (7 lines of code) with functional prototype:

```
int count11(unsigned int n);
```

to return a count of the number of times we find the bottom two bits set in n, starting with the initial n and successively looping to shift n to the right by 1, until n becomes 0. E.g., n = 0xF should return a count of 3: with the bottom 8 bits being successively: 00001111 (count = 1), 00000111 (count = 2), 00000011 (count = 3), 00000001 (still count = 3), and return when n = 0.

II. (36 points) Write a program with main and a function named "exchg", with functional prototype:

```
void exchg(char s[ ], char t[ ]);
```

YOU NEED TO DO EVERYTHING NEEDED FOR A PROGRAM IN A SOURCE FILE, all #declares and #defines and #includes, functional prototypes, etc. The main routine should declare two character strings named s1[] and s2[] with the same dimension, a symbolic constant set to 1000. s1[] should be initialized in its declaration to the char string "I am the first string.", and s2[] should be initialized to "I am the second string." The main function should then call exchg to exchange the two character strings, then print out both strings and terminate.

The function exchg should exchange the character strings s[] and t[] up to the terminal '\0' of each string. The function does NOT know the size of the arrays, and thus **cannot define a local array to hold all of one string or all of the other** — you need to do the exchange one character at a time. You can be sure that each array is large enough to hold the other string. **Be careful to properly handle the case where one string is shorter than the other.** (Ask yourself if it is OK to keep copying each array to the other until the second string terminates.)


```

void exchg(char s[ ], char t[ ])
{
    int i, d, mores, moret;
    for(i = 0, mores = 1, moret = 1; mores || moret; i++) {
        d = s[i]; s[i] = t[i]; t[i] = d;          /* exchange next element of s and t      * /
        if(s[i] == '\0')                          /* as soon as s[ ] ends, set mores to FALSE * /
            mores = 0;
        if(t[i] == '\0')                          /* as soon as t[ ] ends, set moret to FALSE * /
            moret = 0;
    }
    return;                                       /* loop will end when second string s or t ends * /
                                           /* not required when function about to end anyway * /
}

```

Letter Grade Scaling of Numeric Grades

