Section Exercises 2: C basics

Problem 1 : String warmup.

Write a predicate function that takes a string and returns a true or false value indicating whether the string contains any double letters (e.g. he**ll**o, b**oo**k, gra**ss**). You should use the functions contained in the strlib library to help you with this.

Problem 2 : Mode.

The text discusses finding the *mean* and the *median* of an array of integers. The *mode* of a set of numbers is simply the value that occurs most often. For example, in the array

43	65	72	75	79	82	82	84	84	84	86	90	94
0	1	2	3	4	5	6	7	8	9	10	11	12

the mean (or average) is the sum of the numbers divided by the number of entries (1020/13 or 78), the median is the value in the midpoint element (`array`[6] or 82), and the mode is the value which occurs most frequently (in this case, 84).

Write a C function `Mode` with the following prototype:

```
int Mode(int array[], int size)
```

The function takes an integer array and the actual number of elements in the array, and returns the mode of the array. If there is more than one mode (which happens, for example, if two or more different values each occur more frequently than other values but themselves occur the same number of times), your program may return any of those values as the mode. Your program may not assume that the array is initially sorted, but you may sort it as part of the implementation of `Mode` by calling the `Sort` routine. If you get this right away, try to think of a different approach that will do the same thing.

Problem 3 : Quick Pointer Questions.

a) What are the types of the two variables in the following declaration?

```
double *p1, p2;
```

b) What happens if you try to dereference a `NULL` pointer?

c) For any variable `x`, is the expression `*&x` basically a synonym for `x`?

d) For any variable `x`, is the expression `&*x` basically a synonym for `x`?

Problem 4 : General pointer problems

a) Re-write the italicized code to not use [] notation

i)

```
int a[10];
a[4] = 5;
```

ii)

```
int a[5][10];
a[3][4] = 15;
```

iii)

```
int **a;
InitArray(&a);
a[3][4] = 15;
```

Assume that InitArray correctly allocates memory for and initializes the two dimensional array;

b) What is wrong with the following implementation of the Strlib Concat function? What would happen if you tried to use this function in place of the implemented Concat?

```
/* Function: Concat
 * Usage : s = Concat(s1, s2);
 * -----------------------------------
 * This function concatenates two strings by joining them end to end.
 * For example, Concat("ABC", "DE") returns the string "ABCDE"
 */
string Concat(string s1, string s2)
{
       int i, firstStringLen;
       firstStringLen = StringLength(s1);

       for (i=0;i<StringLength(s2);i++)
              s1[firstStringLen + i] = s2[i];
       s1[i] = '\0';
       return s1;
}
```