COP 3502 Quiz #1 Version B (SLMP, Dynamic Memory Allocation) Solutions

1) (8 pts) Assume that the struct `reindeer` had been created by the user and a typedef was set up to refer to the type as `reindeer`. Assume that `reindeer` has an integer field called `antlers`. Write lines of code that do the following:

1) Read in an integer (from standard input) into an integer variable **`n`**, that is already declared.

2) Dynamically allocate memory for an array of size n, storing n variables of type **`reindeer`**. Call the array **`list`**.

3) Manually, using a for loop, for each variable in the array, set its `antlers` field to the value 2.

```
scanf("%d", &n);

reindeer* list = calloc(n, sizeof(reindeer));

for (int i=0; i<n; i++)
    list[i].antlers = 2;
```

Grading: 2 pts scanf, 3 pts either malloc or calloc, 3 pts loop (2 pts first line, 1 pt inside line)

2) (10 pts) Write a function that takes in an array, `array`, its length, `n`, and dynamically allocates **a new array** of size `n`, and fills it with the contents of `array`, in reverse order. So, if the input array stores [3,4,5,1,9], the newly returned pointer will point to a new array storing [9,1,5,4,3]. No changes should be made to the original array.

```
// Pre-condition: array is of length n.
// Post-condition: No change is made to array and a new array of
//   size n is created storing array's contents in reverse and a
//   pointer to this new array is returned.
int* makeReverseArray(int array[], int n) {

    // 4 pts, can use calloc or malloc.
    int* res = calloc(n, sizeof(int));

    // 2 pts for loop, 2 pts for assignment
    for (int i=0; i<n; i++)
        res[i] = array[n-1-i];

    // 2 pts for returning correct pointer.
    return res;
}
```

3) (12 pts) Write a function that takes in an array of strings, `words`, its length, `n`, representing the number of words in the array, and returns a pointer to a newly dynamically allocated string that stores each of the strings in words, concatenated together, all in order. (So if words = {"I","ate","three","cookies"}, the pointer to the string "Iatethreecookies" should be returned.) Allocate exactly the necessary space for the dynamically allocated array.

```
#include <string.h>
char* concatAll(char** words, int n) {

    int len = 0;                                    // 1 pt
    for (int i=0; i<n; i++)                         // 1 pt
        len += strlen(words[i]);                    // 2 pts

    char* res = calloc(len+1, sizeof(char));        // 4 pts

    for (int i=0; i<n; i++)                         // 1 pt
        res = strcat(res, words[i]);                // 2 pts
    return res;                                     // 1 pt

}
```

4) (5 pts) Write a segment of code to free the memory allocated for an array of `n` strings, each of which were dynamically allocated, stored in the array `words`. (Note: The type of `words` is `char**`. You should write 3 lines of code.)

```
    for (int i=0; i<n; i++)             // 1 pt
        free(words[i]);                 // 2 pts
    free(words);                        // 2 pts
```