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

1) (11 pts) Here is the definition for a struct block:

```
typedef struct block {
    int number;
    char letter;
} block;
```

Write a segment of code that (a) reads in an integer into an integer variable `n`, (b) allocates an array of **pointers** to block of size n, and (c) allocates each individual pointer in the array to point to a new block, setting each one's number to a random integer in between 1 and 10 and setting each one's letter to a random letter in between 'a' and 'z'. (Note: given an integer `val` that is in between 0 and 25, (char)(`val`+'a') will be a random letter in between 'a' and 'z'.)

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

block** arr = calloc(n, sizeof(block*));

for (int i=0; i<n; i++) {
    arr[i] = malloc(sizeof(block));
    arr[i]->number = rand()%10 +1;
    arr[i]->letter = (char)(rand()%26+'a');
}
```

Grading: 1 pt scanf, 3 pts either malloc or calloc, 1 pt loop, 2 pts malloc, 2 pts number, 2 pts letter

2) (9 pts) Write a function that takes in an array, `array`, its length, `n`, and an integer `add`, and dynamically allocates **a new array** of size `n`, and fills it with the contents of `array`, but adding `add` to each corresponding item. So, if the input array stores [3,4,5,1,9], and add equals 4, the newly returned pointer will point to a new array storing [7,8,9,5,13]. 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, with add added to
//   each corresponding value.
int* makeElevatedArray(int array[], int n, int add) {

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

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

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

3) (10 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 the first character of each word, all in order. (So if words = {"who","ate","three","cookies","happily"}, a pointer to the string "watch" should be returned.) Allocate exactly the necessary space for the dynamically allocated array.

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

    // 4 pts, can use calloc or malloc.
    char* res = malloc((n+1)*sizeof(char));

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

    // 1 pt for assigning this.
    res[n] = '\0';

    // 1 pt for returning correct pointer.
    return res;

}
```

4) (5 pts) Assume that `array` is an integer array of size `n`. The line of code below reallocates the amount of space for the array to store `2n` integers. Write equivalent code (to the realloc line) that uses the malloc and free functions (so it will be several lines long), assuming that the malloc succeeds in finding the requisite memory.

```
array = realloc(array, 2*n*sizeof(int));
```

```
int* tmp = calloc(2*n, sizeof(int));        // 1 pt
for (int i=0; i<n; i++)                     // 1 pt
    tmp[i] = array[i];                      // 1 pt
free(array);                                // 1 pt
array = tmp;                                // 1 pt
```