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

1) (6 pts) Write several lines of code which do the following: (1) read in an integer from the user and store it in a variable called n, (this is two lines) (2) dynamically allocate an array of n ints called nums, (3) set each of those values to 0.

```
int n;
scanf("%d", &n);
int* nums = calloc(n, sizeof(int));
```

Grading: 1 pt declare n, 1 pt scanf(must have & to get the pt), 4 pts last line (1 pt LHS, 1 pt calloc, 1 pt n, 1 pt sizeof(double))

2) (12 pts) Edit the binary search code below so that it returns the number of while loop iterations necessary before finding value in the array list. If value is not in list, -1 should be returned.

```
// Pre-condition: list1 is length n, sorted, with unique values.
// Post-condition: If value is in list, then the # of while loop
//                 iterations run is returned. Otherwise -1 is returned.
int search(int list[], int n, int value) {

    int low = 0, high = n-1, res = 1;

    while ( low<=high  ) {              // 2 pts

        int mid = (low+high)/2 ;        // 2 pts

        if (value < list[mid]) {

              res++  ;                  // 2 pts

              high = mid-1 ;            // 2 pts
        }
        else if (value > list[mid]) {

              res++  ;                  // 2 pts

              low=mid+1 ;               // 2 pts
        }
        Else                        // Note: ans 3,4 can be swapped
            return res ;            // so can ans 5,6
    }
    return -1;
}
```

3) (12 pts) We can create a struct to store a fraction by having two integer components for the numerator and denominator. Write a function that dynamically allocates and array of pointers to fractions, and then dynamically allocates **n** fractions and makes each of those fractions random values in between 0 and 1, inclusive. To do this, for each fraction, pick the denominator to be a random integer between 1 and **max**. Then, pick the numerator to be in between a random integer in between 0 and the denominator just chosen. Write a function to create these fractions and return a pointer to the array of pointers. (Note: the rand() function returns a random integer in between 0 and 32,767 for the purposes of this problem and you may assume the random number generator has been seeded already.)

```
typedef struct fraction {
    int x;
    int y;
} fraction;

// Pre-conditions: 0 < max < 32767, 1 < n < 1000000
fraction** makeRandList(int n, int max) {

    fraction** res = malloc(n*sizeof(fraction*));      // 3 pts

    for (int i=0; i<n; i++) {                          // 1 pt

        res[i] = malloc(sizeof(fraction));             // 2 pts
        res[i]->y = 1 + rand()%max;                    // 2 pts
        res[i]->x = rand()%(res[i]->y+1);              // 3 pts
    }

    return res;                                        // 1 pt

}
```

4) (5 pts) Let **fracList** be the pointer to the array of pointers to fractions in main with length **n**. Write the necessary code to free all of the memory allocated by the makeRandList function.

```
for (int i=0; i<n; i++)                // 1 pt
    free(fracList[i]);                 // 3 pts
free(fracList);                        // 1 pt
```