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

1) (6 pts) Dynamically allocate an integer array, **values,** of size 1000000 and fill each index of the array with the value of the units digit of that index. (For example, values[1245] should be set to 5. Note that you must set all 1000000 values in the array.)

```
int* values = malloc(sizeof(int)*1000000);        // 2 pts
for (int i=0; i<1000000; i++)                     // 2 pts
    values[i] = i%10;                             // 2 pts
```

2) (12 pts) Complete the function below so it that takes in a sorted array, list, of size n, an integer, max, and creates a dynamically allocated integer array which stores the values from the original array list that are no bigger than max, and returns a pointer to that array. (In addition, the function sets *news to the length of the returned array.) **You are guaranteed that at least one number in the array less than or equal to max.**

```
// Pre-condition: list is length n, sorted, with unique values.
//                list[0] <= max
// Post-condition: All elements in list less than or equal to max
//                 are stored in an array and a pointer to that
//                 array is returned. In addition, *newlen is set to
//                 the length of the returned array.
int* nomorethanlist(int list[], int n, int max, int* newlen) {

    int i=0;
    while ( i<n && list[i]<=max ) i++ ;  // 2 pts, 2 pts, 2 pts


    int* res = malloc(i*sizeof(int));    // 3 pts


    for (int z=0; z<i; z++)

        res[ z ] = list[ z ];            // 1 pt, 1 pt

    *newlen = i ;                        // 1 pt
    return res;
}
```

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
```