ENSF 3378 Tutorial 5 – Oct 17, 2018

Problem I – Assuming all header files are included, draw a memory diagram for point one:

```
#define SIZE 5
int array [SIZE] = {1, 2, 3, 4, 5};

typedef struct vector{
    int* storage;
    int length;
} Vector;

void populate(Vector *p, int n){
    for(int i = 0; i < n; i++)
        for(int j = 0; j < p[i].length; j++)
            p[i].storage[j] = array[j] + j + i;
    // point one
}

int main(){
    Vector *p;
    p = malloc(sizeof(Vector) * 2);
    assert(p != NULL);

    for(int i =0; i < 2; i++) {
        p[i].storage = malloc(SIZE * sizeof(int));
        assert(p-> storage != NULL);
        p[i].length = SIZE;
    }
    populate(p, 2);
    Vector q = *(p+1) ;
    // point 2
    printf("%d  %d", q.storage[1], q.length);
    return 0;
}
```

Problem II: Consider the definition of the following structures and write the definition of function `create_array`.

```
typedef struct Point { double x, y; } Point;

typedef struct Circle{
    double radius;
    Point* center;
} Circle;

Circle* create_array(int n);
/* REQUIRES: n > 0
 * PROMISES: creates and array of Circle with n elements on the
 *           heap
 */
```

Problem III: Consider the definition of the given structures in problem II, and write the definition of function `resize`. You are **NOT** allowed to use library function `realloc`.

```
Circle* resize(Circle* arr, int oldsize, int newsize);
/* REQUIRES: arr points to an existing array of circles with
             oldsize elements
   PROMISES: arr to point to an array with newsize elements and
             preserves the values in the old array up to the
             lesser of the newsize or the oldsize
 */
```