## sort.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Sort an array A using insertion sort.
// Notice it is passed by reference.
void sort(int *A, int array_size) {
        int cur_elem, insert_index;
        for (int cur_index = 1; cur_index < array_size; cur_index++) {
                cur_elem = *(A+cur_index);
                insert_index = cur_index - 1;
                // For each element in A, search for where it belong in the subarray
preceeding it's current location
                while (insert_index >= 0 && cur_elem < *(A+insert_index)) {
                        *(A+insert_index+1) = *(A+insert_index);
                        insert_index -= 1;
                }
                *(A+insert_index+1) = cur_elem;
        }
}

int main() {
        // Allows us to generate random numbers
        srand(time(NULL));

        // Read a user input integer and store it in n
        int n;
        printf("Enter an integer n: ");
        scanf("%d", &n);

        // An array named array. Change this to a dynamic array using malloc.
        int *array = malloc(n * sizeof(int));

        // Assign each element in the array a random number between 0 and 31,999
        for (int i=0; i<n; i++) {
                *(array+i) = rand() % 32000; // assigns random numbers
        }

        // Prints out the elements of the unsorted array
        printf("The unsorted array is: ");
        for (int x = 0; x < n; x++) {
                printf("%d ", *(array+x));
        }
        printf("\n");

        // Calls the sort function to sort the array
        sort(array, n);

        // Print out the elements of the now (supposedly) sorted array.
        printf("The sorted array is: ");
        for (int x = 0; x < n; x++) {
                printf("%d ", *(array+x));
        }
        printf("\n");

        free(array); //Making sure to free what I malloc!
```

```
        return 0;
}
```

## resize.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int main() {
        // Allows you to generate random number
        srand(time(NULL));

        // Allows user to specify the original array size, stored in variable n1.
        printf("Enter original array size: ");
        int n1 = 0;
        scanf("%d", &n1);

        // Create a new array of n1 ints
        int *a1 = malloc(n1 * sizeof(int));

        for(int i = 0; i < n1; i++) {
                // Set each value in a1 to 100
                *(a1+i) = 100;

                // Print each element out (to make sure things look right)
                printf("%d ", *(a1+i));
        }

        // User specifies the new array size, stored in variable n2.
        printf("\nEnter new array size: ");
        int n2 = 0;
        scanf("%d", &n2);

        // Dynamically change the array to size n2
        a1 = realloc(a1, n2* sizeof(int)); // Resize the array

        // If the new array is a larger size, set all new members to 0.
        // Reason: dont want to use uninitialized variables.

        if (n2 > n1) {
                for(int j = n1; j<n2; j++) {
                        *(a1+j)=0;
                }
        }

        for (int i = 0; i < n2; i++) {
                //Print each element out (to make sure things look right)
                printf("%d ", *(a1+i));
        }
        printf("\n");

        // Done with array now, done with program :D
        free(a1);

        return 0;
}
```

# 6.S096 Introduction to C and C++

IAP 2013