```c
#include "vector.h"

// Alumno: Daniel Bizari
// Padrón: 100445
// Correctora: Camila Dvorkin

// Funciones del alumno
void vector_destruir(vector_t* vector){
    free(vector->datos);
    free(vector);
}

bool vector_obtener(vector_t* vector, size_t pos, int* valor){
    if(vector == NULL) return false;
    if(pos > (vector->tam - 1) || vector->tam == 0) return
    false;

    *valor = vector->datos[pos];
    return true;
}

bool vector_guardar(vector_t* vector, size_t pos, int valor){
    if(vector == NULL) return false;
    if(pos > (vector->tam - 1) || vector->tam == 0) return
    false;

    vector->datos[pos] = valor;
    return true;
}

size_t vector_largo(vector_t* vector){
    if(vector == NULL) return 0;

    return vector->tam;
}

// Funciones implementadas por la catedra.
vector_t* vector_crear(size_t tam) {
    vector_t* vector = malloc(sizeof(vector_t));

    if (vector == NULL) {
        return NULL;
    }
    vector->datos = malloc(tam * sizeof(int));

    if (tam > 0 && vector->datos == NULL) {
        free(vector);
```

```c
        return NULL;
    }
    vector->tam = tam;
    return vector;
}

bool vector_redimensionar(vector_t* vector, size_t tam_nuevo) {
    int* datos_nuevo = realloc(vector->datos, tam_nuevo *
    sizeof(int));

    // Cuando tam_nuevo es 0, es correcto si se devuelve NULL.
    // En toda otra situación significa que falló el realloc.
    if (tam_nuevo > 0 && datos_nuevo == NULL) {
        return false;
    }

    vector->datos = datos_nuevo;
    vector->tam = tam_nuevo;
    return true;
}

```