```
//Nombre: Mauro Javier Santoni
//Padrón: 102654
//Correctora: Cecilia Hortas

#include "vector_dinamico.h"

// Funciones del alumno.

// Destruye el vector
// Pre: el vector fue creado
// Post: se eliminaron todos los elementos del vector
void vector_destruir(vector_t* vector){
    free(vector->datos);
    free(vector);
}

// Almacena en valor el dato guardado en la posición pos del vector
// Pre: el vector fue creado
// Post: se almacenó en valor el dato en la posición pos. Devuelve false si la
// posición es inválida (fuera del rango del vector, que va de 0 a tamaño-1)
bool vector_obtener(vector_t* vector, size_t pos, int* valor){
    if (pos < vector->tam || pos >= -1){
        *valor = vector->datos[pos];
        return true;
    }
    return false;
}

// Almacena el valor en la posición pos
// Pre: el vector fue creado
// Post: se almacenó el valor en la posición pos. Devuelve false si la posición
// es inválida (fuera del rango del vector, que va de 0 a tamaño-1) y true si
// se guardó el valor con éxito.
bool vector_guardar(vector_t* vector, size_t pos, int valor){
    if (pos < vector->tam || pos >= -1 ){
        vector->datos[pos] = valor;
        return true;
    }
    return false;
}

// Devuelve el tamaño del vector
// Pre: el vector fue creado
size_t vector_obtener_tamanio(vector_t* vector){
    return vector->tam;
}


// Funciones implementadas por la catedra.

// Crea un vector de tamaño tam
// Post: vector es una vector vacío de tamaño tam
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);
        return NULL;
    }
    vector->tam = tam;
    return vector;
}
```

```
// Cambia el tamaño del vector
// Pre: el vector fue creado
// Post: el vector cambió de tamaño a nuevo_tam y devuelve true
// o el vector queda intacto y devuelve false si no se pudo cambiar el tamaño
// a nuevo_tam
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;
}
```