```c
//gcc -Wall -g -o exe2 exe2.c
// Spinelli Isaia
// Exercice 2 corrigé (Labo Valgrind)

#include "assert.h"
#include "stdbool.h"
#include "stdio.h"
#include "stdint.h"
#include "stdlib.h"
#include <string.h>
// An arbitrary pointer to represent an element in the vector.
typedef void* element_t;

// An expandable array of element_ts.
typedef struct vector_t {
  size_t length;
  element_t *arry;
} *vector_t;

// On success, return vector_t with an initial length of n.
// On failure, returns NULL.  Assumes v != NULL.
vector_t VectorCreate(size_t n);

// Frees the memory allocated for the vector_t.  Assumes v != NULL.
void VectorFree(vector_t v);

// Sets the nth element of v to be e.  Returns the previous nth element_t in prev.
// Freeing e is the responsibility of the user, not the vector_t.
// Returns true iff successful.  Assumes v != NULL.
bool VectorSet(vector_t v, uint32_t index, element_t e, element_t *prev);

// Returns the element at the given index within v.  Assumes v != NULL.
element_t VectorGet(vector_t v, uint32_t index);

// Returns the length of v.  Assumes v != NULL.
size_t VectorLength(vector_t v);

//// Helper functions (assume not buggy)

// Returns a copy of arry with new length newLen.  If newLen < oldLen
// then the returned array is clipped.  If newLen > oldLen, then the
// resulting array will be padded with NULL elements.
static element_t *ResizeArray(element_t *arry, size_t oldLen, size_t newLen);

// Print the elements in the vector on a line.
static void PrintIntVector(vector_t v);

#define N 10 // Test vector length.
int main(int argc, char *argv[]) {
  uint32_t i;
  vector_t v = VectorCreate(4);

  if (v == NULL)
    return EXIT_FAILURE;

  for (i = 0; i < N; ++i) { // Place some elements in the vector.
    int *x = (int*)malloc(sizeof(int));
    // modif 2
    *x = 0;
    element_t old;
    VectorSet(v, i, x, &old);

  }

  PrintIntVector(v);

  // modif 3
  VectorFree(v);

```

```
70    return EXIT_SUCCESS;
71  }
72
73
74  vector_t VectorCreate(size_t n) {
75    vector_t v = (vector_t)malloc(sizeof(struct vector_t));
76    v->arry = (element_t*)malloc(n*sizeof(element_t));
77    // modif 1
78    v->length = n;
79    if (v == NULL || v->arry == NULL)
80      return NULL;
81
82    return v;
83  }
84
85  void VectorFree(vector_t v) {
86    assert(v != NULL);
87
88    // modif 5
89    for (int i = 0; i < VectorLength(v); ++i)
90        free(v->arry[i]);
91
92    // modif 4
93    free(v->arry);
94    free(v);
95  }
96
97  bool VectorSet(vector_t v, uint32_t index, element_t e, element_t *prev) {
98    assert(v != NULL);
99
100       // ajout
101       //printf("length = %ld - %ld", v->length, VectorLength(v));
102
103   if (index >= v->length) {
104     size_t newLength = index+1;
105
106     v->arry = ResizeArray(v->arry, v->length, newLength);
107     v->length = newLength;
108   } else {
109     prev = v->arry[index];
110   }
111
112   v->arry[index] = e;
113
114   return true;
115 }
116
117 element_t VectorGet(vector_t v, uint32_t index) {
118   assert(v != NULL);
119   return v->arry[index];
120 }
121
122 size_t VectorLength(vector_t v) {
123   assert(v != NULL);
124   return v->length;
125 }
126
127 static element_t *ResizeArray(element_t *arry, size_t oldLen, size_t newLen) {
128   uint32_t i;
129   size_t copyLen = oldLen > newLen ? newLen : oldLen;
130   element_t *newArry;
131
132   assert(arry != NULL);
133
134   newArry = (element_t*)malloc(newLen*sizeof(element_t));
135
136   if (newArry == NULL)
137     return NULL; // malloc error!!!
138
```

```
139     // Copy elements to new array
140     for (i = 0; i < copyLen; ++i)
141       newArry[i] = arry[i];
142
143     // Null initialize rest of new array.
144     for (i = copyLen; i < newLen; ++i)
145       newArry[i] = NULL;
146
147
148     // modif 6
149     free(arry);
150     return newArry;
151   }
152
153   static void PrintIntVector(vector_t v) {
154     uint32_t i;
155     size_t length;
156
157     assert(v != NULL);
158
159     length = VectorLength(v);
160
161     if (length > 0) {
162       printf("[%d", *((int*)VectorGet(v, 0)));
163
164       for (i = 1; i < VectorLength(v); ++i)
165         printf(",%d", *((int*)VectorGet(v, i)));
166
167       printf("]\n");
168     }
169   }
170
```