LAB 9 — Pointers to Structures and Dynamic Memory Allocation

1. Specification

The extendable array data structure allows the array capacity to be increased or reduced according to the current array utilization. We implement a simple extendable array data structure in which elements are inserted at and removed from only one end of the array (the rear in this exercise). Write a C program to implement the insertion and deletion operations, extending or shrinking the array as needed.

2. Implementation

- The codes to be submitted are `util.c` and `util.h`. **Use the given template `util.c` and `util.h`** and fill in yourcode. Submit your work with the following command:
  submit 2031 lab9 util.c util.h.
- You are also given a file named `lab9.c` to test your code. Do not submit file `lab9.c`.
- The first function to be implemented is `insertLast()`. See file `util.c` for its specification. The new element is to be inserted at the rear of the extendable array. When a new element is inserted into a ***full*** array, extend the array by doubling its current capacity C (e.g., if C == 4 then C is increased to 8). Use function `malloc` or `calloc` **only** to allocate memory for an array. Do not use the **realloc** function. Allocate a new array; copy the content of the old (smaller) array to the new (bigger) array; free the old array.
- The second function to be implemented is `removeLast()`. See file `util.c` for its specification. The function removes and returns the last element of the array (i.e., the element that was inserted last). If the array is empty, the function calls function `printErr()` to display an error message and returns -1. After a deletion, if the number of elements in the array falls below C/4 (size < C/4), shrink the array by half of the current capacity, but the minimum capacity should always be 4 (e.g., if C == 16 and size < 4 then C is reduced to 8). Again, use function `malloc` or `calloc` to allocate memory **only** for a new array as explained above. Do not use the **realloc** function.
- You may define your own variables inside functions `insertLast()`and `removeLast()`.
- You need to implement the contents of `util.h`.
- The code should be compiled using:

```
gcc lab9.c util.c –o lab9
```

- In file `util.c` you are given several utility functions such as `initArr()`,`printErr()`and `printArray()`. DO NOT modify these functions.
- Do not modify the function and structure definitions in file `util.c`.

3. Sample Inputs/Outputs

See file `lab9out.txt` for the output from running programs `lab9`.

Common Notes

- Complete the header in file `util.c` and `util.h` with your student and contact information.
- Assume that all inputs are valid. No error checking is required on inputs.