Homework 2

Computer Architecture I ShanghaiTech University
HW1 (in Autolab) HW2 HW3

Task 1: Vector implementation in C

Instructions

Implement a vector in C. See the header file vector.h for details. Implement the code in a file called "vector.c".

Your submission should only include the source code for the vector, `vector.c`. Any other file found will directly result in a grade of 0. We will **NOT** be using your copy of the header file or Makefile.

Compilation of the source code requires no errors or warnings and conformity to the C89 standard. The Makefile syntax is given below.

```
CC=gcc
CFLAGS=-Wpedantic -Wall -Wextra -Werror -std=c89

vector.o: vector.c vector.h
	${CC} ${CFLAGS} vector.c -o vector.o
```

The above compilation requires a main function. However, your submission should **NOT** include any main function.

We will test your program using the functions declared in the header file provided as well as some other functions defined by us. The output of the individual test cases will **NOT** be given in auto-grader and any attempt at retrieving the test cases or the program outputs will be seen as plagiarism.

Consider Learning Before You Start

- Pointers and References
- C Memory Management
- Function Pointers
- Vector
- Iterators

Implementation details

You are required to implement a vector with the following rules:

- When your vector is full, gain its capacity to **VECTOR_GROWTH_FACTOR** * original_capacity
- When the user inputs a NULL pointer where there shouldn't be a NULL pointer, your program has to return VECTOR_ERROR, a NULL pointer, an Iterator with pointer NULL and size 0, etc., but it can never crash.

- We require that you will have one line with comments for every four non-blank lines. Comments have to be meaningful and in English.
- No memory leaks are allowed. Memory leaks will be automatically detected and manually screened and will result in a decrease in your score after the deadline.
- We will also impose a limit of 400 on your LoC (lines of code). Do not exceed this limit as we will deduct points for those who exceed this limit.
- You can implement some helper functions in your code. However, your helper functions should not be accessed out of your file. We will check this manually and deduct points if you failed to do this.

Submission

Refer to the submission section at the end.

Task 2: Libraries

For all tasks assume that `vector.h` will be provided in the working directory and your `vector.c` will be copied to the current working directory.

Task 2.1: Static Library

Instructions

Write a shell script named `staticlib.sh` that when invoked, produces a static library named `libvector.a` from the source files `vector.c` and `vector.h`.

Then create a static-linked executable `staticvector` from `libvector.a` and `test.c`. Do not execute the program in the shell script.

You have to use the command `ar` to create the static library. Then use the command `ld` to create the executable file.

Task 2.2 Dynamic-link Library

Instructions

Write a shell script named `dynamiclib.sh` that when invoked, produces a dynamic-linked library named `libvector.so` from `vector.c`.

Then generate an executable named `dynamicvector` using `libvector.so` as well as `test.c`. Use `ld` for that. Do not execute the program in the shell script.

Submission

You should submit a compressed file named as `hw2.tar` to autolab.

The directory tree of your submission should look like the following :

```
├── vector.c
├── dynamiclib.sh
└── staticlib.sh
```

Note: Autolab grading results:

- `1-x` stands for grading results for task1.
- `2-x` stands for grading results for task2.
- `3-0` stands for grading results for memory leak detection.

Note: your submission should **NOT** contain `main()` function

Test Environment

The test environment on autolab is **Ubuntu 16.04** with **gcc 5.x**

Last Modified: Mar. 2nd, 2020