Contents

Chapter I

Foreword

Your name is KANAYA MARYAM.

You are one of the few of your kind who can withstand the BLISTERING ALTERNIAN SUN, and perhaps the only who enjoys the feel of its rays. As such, you are one of the few of your kind who has taken a shining to LANDSCAPING. You have cultivated a lush oasis around your hive, and in particular, you have honed your craft through the art of TOPIARY, sculpting your trees to match the PUFFY ORACLES from your dreams. You have embraced the tool of this trade, which conveniently is the weapon of choice for those who would hunt the HEINOUS BROODS OF THE UNDEAD which crawl from the sand at sunrise to feast on the light and the living.

It would be convenient if you actually hunted them, but it is of course far too dangerous, every bit as suicidal as attempting to poach the terrible MUSCLEBEASTS who roam at night. So you indulge in your bright fascination with the grim through literature. Just before the sun goes down and you join your flora in rest, you immerse yourself in tales of RAINBOW DRINKERS and SHADOW DROPPERS and FORBIDDEN PASSION.

You are one of the few of your kind with JADE GREEN BLOOD. As such you are one of the few who could be selected and raised by a VIRGIN MOTHER GRUB, an event so rare as to elude documented precedent. She would defend you from desert threats, and though her life would be short, in time you would assure her of progeny.

You are one of the few of your kind whose affection for the aesthetic strongly overpowers instinctive regard for the utilitarian. As such, you are one of the few of your kind who has developed a zeal for FASHION and DESIGN and LIVELY COLORFUL PATTERNS. You decorate your hive with FLORA and FABRIC, as delicately or aggressively as inspiration demands. You are a SEAMSTRESS or a RAGRIPPER or a TREETRIMMER or a LUMBERJACK, whichever you care to be, and your unique hive is equipped with a great supply of advanced technology to accommodate your interests. The technology and indeed the hive itself were all recovered from the ruins nearby when you were very young. The seed of your hive was deployed on the volcanic rocks beneath the sand with the assistance of your lusus and her remarkable burrowing skills, and you have lived there happily together since.

You know the ruins and the hive and everything here that is not sand and rock originated from the world of your dreams. You also know that one day you will visit this world while you are awake. That day is today.

Your trolltag is grimAuxiliatrix and you Tend To Enunciate Each Word You Speak Very Clearly And Carefully

What will you do?

Chapter II

Introduction

So you've learned how to make linked lists and binary trees. You're proud of yourself.

Time to forget about them, and never use them again. In fact, accept that both of these are terrible. Any data structure implementation that calls malloc at every insert is terrible. No exceptions.

You will implement 5 different data structures, for various different purpose.

You will have access to a rudimentary test suite that will be used to help you write working code and to correct it during your evaluation.

Because of the provided test suite, your implementations will have to follow a strict API, provided to you as a header file. You will have to figure out what most of the functions do, by comparing them to existing data structure libraries, and by reading the test file.

Chapter III

Goals

Your project, once finished, will not just have been an introduction to data structures, but also powerful library usable in any 42 C projects, while drastically reducing the amount of copy-pasted code, such as sorts, reallocs, and inserts.

It will allow you to think more modularly, and not reinvent the wheel at every turn.

If you do your job well, you'll have a safe library of multi-purpose tools for any situation.

You will also have to learn how to use the debugging tool of your choice.

Chapter IV

General instructions

This is a C, Norme compliant library project.

The library should be organized in a seperate repository, containing its Makefile and headers, just like your libft. The repository should be named `data` and be placed in your libft or root repository.

The library makefile will produce a `libdata.a` or `libftdata.a` file.

The project Makefile will join said library and the provided main.c file.

Do not include the main.c file in your final commit.

The whole program must compile without warnings or errors.

Both Makefiles must only compile what is necessary, and implement the usual rules.

The maximum allowed optimisation level is -O2.

You will only have access to these functions:

- xmalloc
- xfree
- write

`xmalloc and xfree are defined in the main file, and allow the testing suite to measure your memory performances. As such, calling the real malloc anywhere in your implementations will result in a failing grade. If the call was not an accident, but an attempt to trick the memory counter, the corrector may decide to give you a cheating grade of -42.`

Your implementations must not crash or cause leaks during valid use. Valid use is defined as:

- anything in the main.c file
- anything that a normal user might reasonably do

As such, should the executable built with the provided main file show any leaks, or crash at any time, you will fail the project. The only exception is a potential error in the testing suite, which should be reported to me immediatly.

Correctors will be encouraged to write their own tests, and use the manual tests provided.

You must provide tests for any bonus beyond the suggested ones.

You must have an auteur file in your root repository. It can however contain anything you want.

Crashes, leaks and memory corruptions are allowed only if the alternative would come at an unreasonable cost is performance, in API simplicity or in freedom in use. See "iterator invalidation" in c++.

```
Should you hesitate between safe and fast, usually choose safe.
```

Should you wish to change the main file, for a SIMPLE and REASONABLE change, which DOESN'T give you an unfair advantage on ANY part of the project, you main join a diff file to be applied to the main during evaluation.

```
Lookup "debugging c in vscode", malloc_usable_size and
-fsanitize=address for help during development.
```

Chapter V

Mandatory part

Fully implement the API as used by the main.c with the flags:

- TEST_ARRAY
- TEST_BITSET
- TEST_QUEUE
- TEST_SORTED
- TEST_PMA

V.1 The dynamic array

Excidingly boring, but extremely useful. Your implementation should mirror the c++ vector. Your growth factor should be defined in a constant, to be easily modifiable if you want to prioritize memory or performance.

V.2 The bitset

Because who can say no to reducing your memory overhead by 8?
It is inspired by the bitset implementation of the Rust and c++ standard libraries.
Again, the growth factor should be defined as a constant.

V.3 The double ended queue

Inspired by the `dequeue` implementation in Rust.
You will very probably have to implement it with a ring buffer.

V.4 The sorted array

A bit exotic, and not very fast, but a good introduction for the following part. Allows for incredibly easy "insert sort" and search.
You will not allow duplicate elements by default, but may add it as a bonus.

V.5 The packed memory array

Like the btree, binary tree and hashtable, it allows searching for elements in log(n) time. Unlike the hashtable, it can iterate on elements in a specific order. Unlike the binary tree, it is FAST. Unlike the btree, it is not too hard to implement. The API is close to a btreemap in rust or a map in c++.

Google each of those containers, ask yourself "can I do that", then think again.

None of the pointers you receive are meant to be stored directly inside the structure. The reason you receive a size parameter is to copy the memory.

During debug, check the state of your structures every call, using malloc_usable_size.

Reimplement realloc, and zero-init all allocated memory. It will make your life much easier.

Chapter VI

Bonus part

VI.1 Bonuses A: Tested Bonuses

Fully implement 5 of the 6 bonus functions sets as used by the main.c with the flags:

- TEST_ARRAY_BONUS
- TEST_BITSET_BONUS
- TEST_SORTED_BONUS
- TEST_PMA_BONUS_IT
- TEST_PMA_BONUS_IT_BACK
- TEST_PMA_BONUS_MULTI

VI.2 Bonuses B: Custom Bonuses

Some suggestions for your own bonuses:

- A define to switch between to switch between the real `realloc` and your implementation.
- Additional functions such as array_concat, or pma_filter
- Making a template system to generate a data structure specialized for any type
- Using macros to automatically choose the right size for various functions
- Using the _Generic macro to check that the right type was inserted
- Using a canary system to check that iterators are still valid when used

`Custom bonuses won't be graded unless the tested part is complete`

Chapter VII

Turn-in and peer-evaluation

Turn in your work using your `GiT` repository, as usual. Only the work that's in your repository will be graded during the evaluation.