# Project 2

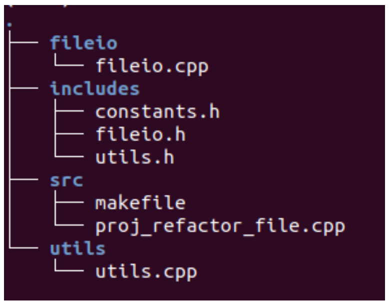
Your task is to break apart a monolithic file into related modules in a sensible directory structure. Please leave all functions as written (that is unimplemented).

This project has several parts;

1. Refactoring a monolithic cpp file into smaller modular files and directories
2. Using #includes effectively
3. Using relative paths
4. Writing a makefile to accompany your newly modularized solution.

This project does NOT use eclipse.All you need is an editor (VSCode, Sublime, micro, nano...whatever you like) and the GNU toolchain.

**Refactoring a monolithic cpp file into smaller modular files and directories**
You are given proj_refactor_file.cpp.  In it there is a collection of constants and functions.  Please break this file up into the following directory structure;



```
.
├── fileio
│   └── fileio.cpp
├── includes
│   ├── constants.h
│   ├── fileio.h
│   └── utils.h
├── src
│   ├── makefile
│   └── proj_refactor_file.cpp
└── utils
    └── utils.cpp
```

fileio.cpp: contains load and save function definitions and all necessary includes
fileio.h: contains load and save function declarations and all necessary includes

utils.cpp: contains sort, getNext, getSize and handleMissingData function definitions and all necessary includes
utils.h: contains sort, getNext, getSize and handleMissingData function declarations and all necessary includes

constants.h: contains all constants, the enum and the struct and all necessary includes

proj_refactor_file.cpp: contains only main() and all necessary includes after you move everything else

makefile: used to build and clean this project (see starter makefile)

**Using #includes effectively**

- For all of the above files, use the minimum number of includes needed for the file to compile.
- All .cpp files should #include their associated .h files.
- Please include system includes before user includes
  - *For instance sort in utils.h has a vector that's passed by reference. So you must include vector; like so;*

```
#include <vector>
```

- If a cpp or header file uses a function or constant declared or defined in another .h or .cpp file then it must #include the appropriate .h file.
  - *For instance utils.h declares the function sort which uses SORT_ORDER. SORT_ORDER is defined in constants.h, so utils.h must include constants.h like so;*

```
#include "constants.h"
```

- Please do not #include cpp files! This is an automatic 30 pt deduction
- Please use include guards in all .h files

**Using relative paths**

Please use relative paths in all includes and in the makefile. For instance;
fileio.cpp must include fileio.h. So the following line should appear at the top of fileio.cpp after any system includes;

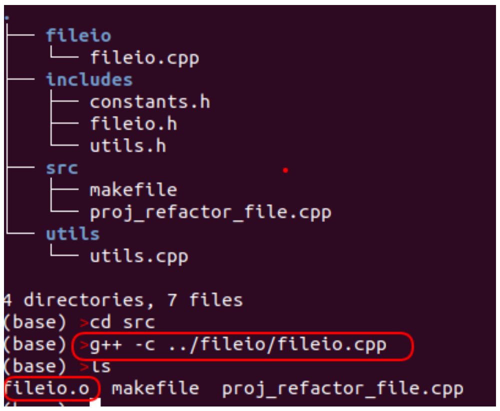
```
#include "../includes/fileio.h"
```

This line includes fileio.h by going up 1 directory (..) then down into the includes directory (/includes) and then selects fileio.h for inclusion.

Please do this as needed.

**Writing a makefile to accompany your newly modularized solution.**

Please complete all TODOs in the skeleton makefile provided. Note GNU make and g++ can process relative paths. For example from the source directory of the completed project I can compile fileio.cpp to fileio.o as shown below.



```
.
├── fileio
│   └── fileio.cpp
├── includes
│   ├── constants.h
│   ├── fileio.h
│   └── utils.h
├── src
│   ├── makefile
│   └── proj_refactor_file.cpp
└── utils
    └── utils.cpp

4 directories, 7 files
(base) >cd src
(base) >g++ -c ../fileio/fileio.cpp
(base) >ls
fileio.o  makefile  proj_refactor_file.cpp
```

Relative paths work the same way in makefiles.

**Grading:**
*(50%) Correctly refactor monolithic file as annotated above*

*(20%) Correctly implement all TODO's in the provided makefile.*

*(20%)Push your solution to a git repository (github, bitbucket, gitlab etc…)*
You should push 7 files to your repository; fileio.cpp, constants.h,fileio.h, utils.h, makefile, proj_refactor_file.cpp and utils.cpp

(10%) Correctly submit a single file to scholar called *submission.txt.* This file should have a single line which is your remote repo address (see below)

**Scholar Submission:**
Blackboard cannot seem to handle a simple URL. So please submit a text file called *submission.txt,* in this file please have a single line with a link to your git repository. My script will open the file, get the link, clone your repo and then test as annotated below. The line in *submission.txt* should look similar to:

https://github.com/CNUClasses/proj2.git

**I will test by;**
Cloning your repository
Running 'make' and verifying correct executable generated
running 'make clean' and verifying that all compile byproducts are removed
Running combinations of your files and my files to verify correct output. Note: I will probably change the values in constants.h when testing your code.

**Potential Gotchas:**
Filenames are case sensitive, F.cpp is not the same as f.cpp
Make sure your git repository is publicly accessible.