VE482 Lab 5 Report

Lan Wang 519370910084

Layer programming

1. **The program can be divided into three layers, what are they?**

   Interface, logic and list.

2. **Split the program into files according to the defined layers.**

3. **Create the appropriate corresponding header files.**

4. **If necessary rewrite functions such that no call is emitted from lower level functions to upper level functions.**

   For 2 and 3 and 4, see:

   Interface: `interface.c`, `interface.h`

   logic : `logic.c`, `logic.h`

   list: `list.c`, `list.h`

5. **The initial program implements a command line interface, write a "Menu interface" which:**

   (i) welcomes the user,

   (ii) prompts him for some task to perform, and

   (iii) runs it.

   When a task is completed the user should:

   (i) be informed if it was successful and then

   (ii) be displayed the menu.

   From the menu he should be able to exit the program.

   See `dispatchMenu()` and related functions in `interface.c`.

6. **Write two main functions, one which will "dispatch" the work to another function which will run the command line user interface and a second one which will "dispatch" the work to the Menu user interface.**

   See `menu.c` and `cmd.c` for menu user interface and command line user interface respectively.

Libraries

1. **What are the four stages performed when compiling a file? Briefly describe each of them.**

   Pre-processing: In this process, comments are removed and macros and included files are expanded.

   Compilation: Transform to assembly language.

   Assembly: Create object file.

   Linking: Link object file and libraries.

2. **Two types of libraries exist: static and dynamic. Explain the difference between the two.**

   Static: Included in program, and needed to be recompiled when the library changes.

   Dynamic: Loaded at runtime, so no need to recompile it when changing its content.

Static libraries

1. **Search more details on how to proceed.**

   With library file `libStatic.a`, source coed `test.c`:

```
gcc -Wall -Wextra -Werror -pedantic -c *.c # get .o file
ar -rc libStatic.a *.o
gcc test.c ./libStatic.a -o exename
```

2. **Create two static libraries, one for each of the two lowest layers in the previous program.**

```
gcc -Wall -Wextra -Werror -pedantic -c *.c
ar -rc libList.a list.o
ar -rc libLogic.a logic.o
```

3. **Compile the command line version of the program using these two static libraries.**

```
gcc cmd.c interface.c interface.h ./libList.a ./libLogic.a -o listCmd
```

Dynamic libraries

1. **Generate two dynamic libraries, one for each of the two lowest layers in the previous program.**

```
gcc -fPIC -Wall -Werror -Wextra -pedantic list.c list.h -shared -o
libList.so
gcc -fPIC -Wall -Werror -Wextra -pedantic logic.c logic.h -shared -o
libLogic.so
```

2. **Compile the whole program**

```
gcc -Wall -pedantic -Werror -Wextra cmd.c interface.c interface.h logic.c
logic.h list.c list.h -o listCmd
gcc -Wall -pedantic -Werror -Wextra menu.c interface.c interface.h logic.c
logic.h list.c list.h -o listMenu
```

3. **Compile the Menu version of the program using these two dynamic libraries.**

```
gcc -Wall -pedantic -Werror -Wextra -L. menu.c interface.c interface.h -
lList -lLogic -o listMenu
```

Extra remarks

1. **What is the difference between a library and the API.**

   Library refers to the code itself, while API refers to the interface.

2. **Implement the API below for the two libraries.**

   Files have been loaded to JOJ.