Exercise 1

- Which of these lines belong into the header file? Why?

```
char ch; declaration
string s; declaration
extern int error_number; extern def.
static double sq(double); Method decl.
int count=1;
const double pi=3.5; // ;) const
struct fraction { int c; int d; };
char *name="It's me";
char *prog[]={"echo","hello","world!",NULL};
extern "C" void c_swap(int *a, int *b); extern / decl.
double sqrt(double);  decl.
void swap(int &a, int &b) {  int c=a; a=b; b=c;  }
namespace NS { int a; }
struct user;  decl.
```

Exercise 2

- Separate compilation
  - Use the fraction program from the last lecture and move the fraction data type into the fraction.cc and fraction.h files respectively
  - Move all utility functions into util.cc and util.h files
  - Move the main program into fraction-test.cc
  - Write a Makefile and compile the above

Exercise 3

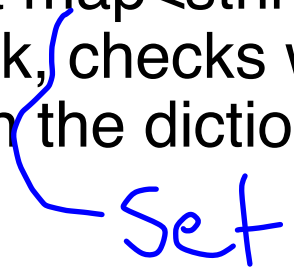
- Implement an RPN (Reverse Polish Notation Calculator)
  - When the user enters ‘q’ the program terminates
  - When the user enters ‘n’ followed by a number, it is put on the stack
  - When the user enters ‘d’ the last number is removed from the stack
  - When the user enters ‘+’, ‘-’, ‘*’, ‘/’, the calculator takes the last two numbers from the stack applies the operation to the numbers and puts the result on the stack
  - Use the std::vector container and its iterator!

Exercise 3 – An Example (user input in bold)

```
Command: n 2 n 4 n 3
1: 2
2: 4
3: 3
Command: *
1: 2
2: 12
Command: -
1: -10
Command: n 2 /
1: -5
Command: d
Command: q
```

Exercise 4

- Simple Spell Checker
  Implement a simple spell checker. The spell checker takes two files as command line arguments, a dictionary file containing a list of correctly spelled words and a file whose content is to be checked. Upon startup, your program stores the words contained in the dictionary file in a ~~map~~<string>. Then it reads every word in the file to spell check, checks whether each word is correctly spelled (ie contained in the dictionary file) and if not displays it on cout.



Exercise 4 – An Example

dict.txt

```
correctly
spelled
words
are
stored
in
the
dictionary
```

text.txt

```
a comprehensive dictionary
is important.
```

```
C:\> check dict.txt text.txt
a
comprehensive
is
important
C:\>
```

Next Lecture

- Object-Based Programming
- Classes
- Templates

Have fun solving the examples!

See you next week!