Exercise on compiling and running C code

This is an easy exercise on writing and compiling a simple C program. If you are new to C or Linux, you are strongly encouraged to do this exercise, as later assessed exercises will use a similar setup. In particular, later exercises will be marked by compiling and running them on the Linux lab machines (which you can access from home via ssh).

Implement the Fibonacci function in a file called `fib.c`. The differences to Java are that there is no `static public` at the beginning of the function, and the function does not require an enclosing object the way a Java method does. So your code should just be

```
long fib(long m)
{
    // the usual code for Fibonacci
    // just like in Java
}
```

To test your code, you will need the files `fibmain.c`, containing a main function, and `fib.h`, which provides the inferface to what you need to implement (which in this case is just the type of the function `fib`). The files are here:

`http://www.cs.bham.ac.uk/~hxt/2016/c-plus-plus/fibmain.c`
`http://www.cs.bham.ac.uk/~hxt/2016/c-plus-plus/fib.h`

On the Linux lab machines, you need to load LLVM to use the Clang C compiler. Type the following in the command line:

```
module load llvm
```

Compile the code by typing the following into the command line, in a directory containing all the above files:

```
clang -Werror -o fibmain fibmain.c fib.c
```

Then run the compiled program using:

```
./fibmain
```

You will see whether your fibonacci function has passed some tests.

Note, however, that the naive recursive implementation is very inefficient. It is much faster to cache the values of the function in an array so they do not have to be recomputed. Write a second implementation that uses an array. In your C code, you should declare the array outside the function, like this:

```
long fibvals[1000]; // array declaration

long fib(long m)
{
    // code for Fibonacci using fibvals above
}
```

Accessing array elements has the same syntax as in Java.