Computing Fibonacci numbers

The Fibonacci numbers (sequence A000045 at the OEIS) are generally defined using the recurrence relation $F_n = F_{n-1} + F_{n-2}$, with $F_0 = 0$ and $F_1 = 1$. Thus, the first ten Fibonacci numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

The most obvious way to compute Fibonacci numbers is recursively, using the recurrence relation directly. Another, more efficient way (an example of "dynamic programming") is to start at the beginning and keep track of the two most recently-computed values, iteratively computing the next until the desired point is reached. Finally, like every sequence defined by linear recurrence, the Fibonacci numbers have a closed-form solution, and F_n can be computed directly by rounding $\frac{\phi^n}{\sqrt{5}}$ (where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio) to the nearest integer.

Define three functions, each of which takes an index n as an argument, and returns F_n, computed recursively, iteratively, or directly with the closed-form solution respectively. Use the following header:

fibonacci.h

```
unsigned int fibonacci_recursive(unsigned int n);
unsigned int fibonacci_iterative(unsigned int n);
unsigned int fibonacci_closed(unsigned int n);
```

Put each function in its own file; for example, implement `fibonacci_iterative` in `fibonacci_iterative.c`. In a file named `fibonacci.c`, write a `main` function that prints out the first ten Fibonacci numbers computed by each function in a table with columns for recursive, iterative, and closed-form, from left to right. Since the functions should obviously all return the same values, the correct output is:

```
$ ./fibonacci
0       0       0
1       1       1
1       1       1
2       2       2
3       3       3
5       5       5
8       8       8
13      13      13
21      21      21
34      34      34
```

If you name your files as indicated, the following `Makefile` will enable you to build your program by simply typing `make`.

Makefile

```
CFLAGS=-Wall -g
LDFLAGS=-lm

OBJECTS=\
  fibonacci.o \
  fibonacci_recursive.o \
  fibonacci_iterative.o \
  fibonacci_closed.o

fibonacci: $(OBJECTS)
fibonacci.o: fibonacci.c fibonacci.h
fibonacci_recursive.o: fibonacci_recursive.c fibonacci.h
fibonacci_iterative.o: fibonacci_iterative.c fibonacci.h
fibonacci_closed.o: fibonacci_closed.c fibonacci.h

.PHONY: clean
clean:
	rm -f fibonacci $(OBJECTS)
```

For the closed-form computation, you will need to cast the `unsigned integer` argument to `double` to maintain the required precision. You will want to know and use the functions `sqrt`, for computing $\sqrt{5}$; `pow`, for computing ϕ^n; and `round`, for correctly rounding the final result to the nearest integer before casting it back to `unsigned int` and returning it. All of those functions are found in `math.h`, and require linking with the math library `libm` (thus the special `LDFLAGS` in the `Makefile`).

None of these functions requires more than a handful of lines of code; although there might be a lot of thinking involved, if you actually find yourself writing a lot of code, perhaps you are approaching the problem the wrong way.

On CMS, submit your source files, `fibonacci.c`, `fibonacci_recursive.c`, `fibonacci_iterative.c`, and `fibonacci_closed.c`; when grading, I will provide the `fibonacci.h` header file and the `Makefile` as they appear above.