Homework 9: Prime Functions

Assigned: Wednesday 01 May
Due Date: Wednesday 08 May

Objectives

More practice implementing functions.

Assignment

This assignment has two parts. In Part 1, you implement a function that checks if a number is prime. In Part 2, your function finds and returns a prime number greater than a given number.

Part 1

In Part 1, your assignment is to implement a function, called `isPrime()`, that checks if the parameter it is given is a prime number. Recall that a number is prime if it is only divisible by 1 and itself. By definition the number 1 is not prime. You can check if n is divisible by m using the modulus operator `%` by checking whether $n\%m$ is zero. So, you simply have to iterate through all possible values of m to determine whether n is prime. (Actually, checking for m up to square root of n is sufficient.) If n is indeed prime, then `isPrime(n)` must return 1. Otherwise, n is composite and `isPrime(n)` must return 0.

Example Compilation and Execution

```
[arsenaul@linux1 hw9]$ gcc -Wall isPrime.c -lm
[arsenaul@linux1 hw9]$ ./a.out
Enter a number: 3
Yes, 3 is prime.
[arsenaul@linux1 hw9]$ ./a.out
Enter a number: 4
No, 4 is not prime.
[arsenaul@linux1 hw9]$
```

Starer Code

```
/****************************************
** File:  isPrime.c
** Author: <myName>
** Date:  <todaysDate>
** Section: CMSC104 Section 02
** E-mail: <myEmailAddress>
**
** This file contains the main program for Part 1 of
** Homework 9.
** The program asks the user for a number and tells
** them if it is prime.
```

```
****************************************/

#include <stdio.h>
#include <math.h>

/* Function prototype */
/* Don't change the function prototype! */

int isPrime(int n) ;

/* Do not change the main program!! */
int main() {
   int n ;

   printf("Enter a number: ") ;
   scanf("%d", &n) ;

   if (isPrime(n)) {
      printf("Yes, %d is prime.\n", n) ;
   } else {
      printf("No, %d is not prime.\n", n) ;
   }
   return 0 ;
}

/* End of main() */

/* Add function comment header here. */

/* Implement your isPrime() function here. */
```

Part 2

In Part 2, your assignment is to write a function `findPrime()` that finds the first prime number larger than the parameter n. Number theory tells us that there is always a prime number between n and $2 \times n$. So, the search will always be successful. The first prime number is 2, so you can exit the function early if the user enters 1. Otherwise, you simply have to iterate through numbers bigger than n until your `isPrime()` function from Part 1 tells you that you have found a prime.

Example Compilation and Execution

```
[arsenaul@linux1 hw9]$ gcc -Wall findPrime.c -lm
[arsenaul@linux1 hw9]$ ./a.out
Enter a number: 2
3 is the first prime number greater than 2.
[arsenaul@linux1 hw9]$ ./a.out
Enter a number: 4
5 is the next prime number greater than 4.
```

```
[arsenaul@linux1 hw9]$
```

Starter Code

```
/****************************************
** File:  findPrime.c
** Author: <myName>
** Date:  <todaysDate>
** Section: CMSC104 Section 02
** E-mail: <myEmailAddress>
**
** This file contains the main program for Part 2 of
** Homework 9.
** The program asks the user for a number and tells
** them the next prime number that comes after it.
****************************************/

#include <stdio.h>
#include <math.h>

/* Function prototype */
/* Do not change the function prototypes */

int isPrime(int n) ;
int findPrime(int n) ;

/* Do not change the main program!! */
int main() {
   int n ;

   printf("Enter a number: ");
   scanf("%d", &n);

   printf("%d is the first prime number greater than n.\n", findPrime(n));

   return 0 ;
}

/* End of main() */

/* Copy your isPrime() function here. */

/* Add function header comment for findPrime() */

/* Implement your findPrime function here. */
```

Extra Credit

+10 points Use separate compilation to organize your program. This means that Part 1 and Part 2 would each have their own `.c` and `.h` files, plus a separate `.c` with the `main()` function. That would mean a total of 6 files, and compiled without errors.

+5 Build on to the separate compilation extra credit by using a Makefile. The Makefile would handle building both Part 1 and Part 2, and compile each `.c` file separately. That would be a total of 6 *targets* in your Makefile, and compile properly without errors. Refer to the Makefile lecture on Blackboard for an example.

Warning If you do the extra credit, be sure to submit the extra files. If you only submit part, and your program doesn't compile, you will lose points.

Grading Rubric

- Part 1 header comment: 2 points
- Part 1 body comments: 3 points
- Part 1 compiles: 20 points
- Part 1 determines prime correctly: 25 points
- Part 2 header comment: 2 points
- Part 2 body comments: 3 points
- Part 2 compiles: 20 points
- Part 2 determines next prime correctly: 25 points
- EC 1: +10 points
- EC 2: +5 points

What to Submit

Use the `script` command to record yourself compiling and running each program 3 times. (Do not record yourself editing your program!) Exit from `script`. Submit your programs and the typescript file. Note: if you are doing the extra credit, there are more files to submit than what is listed below! All files must be submitted for any credit.

```
[arsenaul@linux1 hw9]$ submit cmsc104_arsenaul hw9 isPrime.c findPrime.c typescript
```

Verify Submission

If you *think* you submitted the assignment, but the `submitls` command doesn't show you your file names, then the files were **not** submitted and no grade will be given.

```
[arsenaul@linux1 hw9]$ submitls cmsc104_arsenaul hw9
```

Last modified: 08 February 2023