HOMEWORK 2.3 SAMPLE OUTPUT
==========

```
MS147V:bin tgrushka$ ./factor
Enter a positive integer: 592314
Check for Prime Numbers? 1
Print What (0 = Only Factors, 1 = All Numbers, 2 = Factors + Primes) ?
0
Square Root (rounded down): 770

Number                          Prime  Other Factor
==============================  =====  ==============================
                             1                                 592314
                             2      Y                          296157
                             3      Y                          197438
                             6                                  98719
                            17      Y                          34842
                            34                                 17421
                            51                                 11614
                           102                                  5807

Execution time: 0.000087 s

MS147V:bin tgrushka$
```

```c
/*  HOMEWORK 2.3
factor.c:
Prompts the user for a POSITIVE INTEGER and prints
all the integer factors of that integer.
Tom Grushka
February 6, 2016    */

#define _CRT_SECURE_NO_WARNINGS // allow scanf on Windows
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

/*  Execution time from Thomas Pornin
http://stackoverflow.com/questions/5248915/execution-time-of-c-program  */
clock_t begin, end;
double time_spent;

/*  "dynamic" Array type adapted from casablanca
http://stackoverflow.com/questions/3536153/c-dynamically-growing-array  */
typedef struct {
    long long *array;
    size_t used;
    size_t size;
} Array;

// function to initialise the array
void initArray(Array *a, size_t initialSize) {
    a->array = (long long *)malloc(initialSize * sizeof(long long));
    a->used = 0;
    a->size = initialSize;
}

// function to add an item to the array
void insertArray(Array *a, long long element) {
    if (a->used == a->size) {
        a->size *= 2;
        a->array = (long long *)realloc(a->array, a->size * sizeof(long long));
    }
    a->array[a->used++] = element;
}

// function to free the memory used by the array
void freeArray(Array *a) {
    free(a->array);
    a->array = NULL;
    a->used = a->size = 0;
}
/*  End of Array type   */
```

```
Array prime;    //  define array to store prime numbers

                //  declare functions
long long getLongLong(const char *prompt);
int checkPrime(long long number);

/*  checkPrime: determine if the given number is prime;
if prime, add it to the prime Array and return 1;
if not, return 0.
arguments: number (long long)
returns: integer 0 (false) or 1 (true)  */
int checkPrime(long long number)
{
    int isPrime = 1;    // flag (0 = false, 1 = true)
    long long sq = 0;           // store square root
    sq = (long long)(sqrt(number) + 0.5);   // get square root of number
    if (number == 1) return 0;  // not a prime; prevent false negatives
                                //  these quick checks cut processing time a little
    if (number > 3 && (number % 2 == 0 || number % 3 == 0)) return 0;
    //  loop through the array:
    for (unsigned int i = 0; i < prime.used; i++)
    {
        if (prime.array[i] > sq) break;  // stopping at square root of the number
            saves a whole lot of time
        if (number % prime.array[i] == 0) {
            isPrime = 0;    // divisible by another prime
            break;          // don't check any more
        }
    }
    // if prime, add to array
    if (isPrime) insertArray(&prime, number);
    return isPrime;         // return the flag
}

int main(int argc, char** argv)
{
    long long number = 0;   // input number (long long)
    long long sqroot = 0;   // square root of input number (long long)

    int checkForPrime = 0;      // input - skip prime number checking if 0
    int printAllNumbers = 0;    // input - print out all numbers, not just factors

    int isPrime = 0;
    initArray(&prime, 10);  // initially 10 spaces for prime numbers
    insertArray(&prime, 2);

    // INPUT loop: keep asking for positive integer until we get one
    do {
        number = getLongLong("Enter a positive integer: ");
    } while (number < 1);
```

```
    // INPUT: ask user whether to check for prime numbers
    checkForPrime = (int)getLongLong("Check for Prime Numbers? ");

    // INPUT: ask user whether to print only factors, all numbers, or all primes
    if (checkForPrime)
        printAllNumbers = (int)getLongLong("Print What (0 = Only Factors, 1 = All
Numbers, 2 = Factors + Primes) ?");
    else
        printAllNumbers = (int)getLongLong("Print What (0 = Only Factors, 1 = All
Numbers) ?");

    // calculate and print square root of the number (no need to exceed this)
    sqroot = (long long)(sqrt(number) + 0.5);
    printf("Square Root (rounded down): %lld\n\n", sqroot);

    // print header:
    printf("Number                          Prime  Other Factor\n");
    printf("==============================  =====  ==============================\n");

    begin = clock();    // store start time in order to time execution
                        // begin factor calculation loop
    for (long long factor = 1; factor < sqroot; factor++)
    {
        /*  Call function to check if factor is prime:
        Current implementation requires checking all numbers;
        Possible future improvement.    */
        if (checkForPrime) isPrime = checkPrime(factor);

        // if factor is a factor of number:
        if (number % factor == 0) {
            printf("%30lld  %5s  %30lld\n", factor, (isPrime ? "Y" : ""), number /
factor);
            // otherwise, if we're printing all numbers, or just primes:
        }
        else if (printAllNumbers > 0) {
            if (printAllNumbers == 1 || (printAllNumbers == 2 && isPrime))
                printf("%30lld  %5s\n", factor, (isPrime ? "Y" : ""));
        }
    }

    end = clock();  // set end time, calculate execution time & print:
    time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("\nExecution time: %f s\n\n", time_spent);

    freeArray(&prime);  // clear & free the prime array:

    return 0;   // return without error
}

/*  getLongLong
argument:   prompt (const char *)
```

```
string to prompt the user
return:     user input converted to long long   */
long long getLongLong(const char *prompt)
{
    long long myLongLong = 0;   // long long to return

    printf("%s", prompt);   // Display the prompt
    scanf("%lld", &myLongLong); // request input

    return myLongLong;          // return integer
}
```