HOMEWORK 2.1 SAMPLE OUTPUT
==========================

```
MS147V:bin tgrushka$ ./multiples
Enter an integer greater than zero: 15

Counting to 15 by multiples of 5:
    0 * 5 =       0
    1 * 5 =       5
    2 * 5 =      10
    3 * 5 =      15
MS147V:bin tgrushka$ ./multiples
Enter an integer greater than zero: 47

Counting to 47 by multiples of 5:
    0 * 5 =       0
    1 * 5 =       5
    2 * 5 =      10
    3 * 5 =      15
    4 * 5 =      20
    5 * 5 =      25
    6 * 5 =      30
    7 * 5 =      35
    8 * 5 =      40
    9 * 5 =      45
I cannot reach 47 because it is not a multiple of 5.
MS147V:bin tgrushka$
```

HOMEWORK 2.2 SAMPLE OUTPUT
==========================

```
MS147V:bin tgrushka$ ./lowpower
Good afternoon, Dave. I am a HAL 9000 computer. I am afraid my power needs
adjusting.
Please specify a new voltage and resistance, Dave.
I cannot handle more than 0.25 Watts of power, Dave.
Voltage: 5
Resistance: 1
Power limit exceeded! I'm sorry, Dave. I'm afraid I can't do that.

Please specify a new voltage and resistance, Dave.
I cannot handle more than 0.25 Watts of power, Dave.
Voltage: 5
Resistance: 2
Power limit exceeded! I think you know what the problem is just as well as I
do.

Please specify a new voltage and resistance, Dave.
I cannot handle more than 0.25 Watts of power, Dave.
Voltage: 7.3
Resistance: 253.7

Very good, Dave. That will be 0.21 Watts of power. Have a nice day!
MS147V:bin tgrushka$
```

```
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 2.1
    multiples.c:
    Prompts the user for a POSITIVE INTEGER limit and prints all
    multiples of 5 that DO NOT EXCEED the entered limit.
    Tom Grushka
    February 5, 2016    */

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

const int MULTIPLIER = 5;   // constant multiplier (5 given in assignment spec)

int getInteger(const char *prompt);

int main(int argc, char** argv)
{
    int limit = 0;      // user-specified positive integer to count up to
    int i = 0;          // counter
    int product = 0;    // product = counter * MULTIPLIER

    /*  Input loop: keep prompting user for limit until
        a reasonable value is entered.  */
    do {
        limit = getInteger("Enter an integer greater than zero: ");
    } while (limit <= 0);

    // Tell user what we're doing
    printf("\nCounting to %d by multiples of %d:\n", limit, MULTIPLIER);

    /*  Begin output loop:
        As long as i * MULTIPLIER <= limit ... */
    while (product <= limit) {
        printf("%5d * %1d = %7d\n", i, MULTIPLIER, product);  // pretty print
        i++;    // increment counter
        product = i * MULTIPLIER;   // calculate product
    }   // if while condition not met, stop looping

    if (limit % MULTIPLIER > 0) // check for remainder
        // explain remainder to user
        printf("I cannot reach %d because it is not a multiple of %d.\n", limit,
MULTIPLIER);

    return 0;   // We shouldn't have an error
}

/*  getInteger
    argument:   prompt (const char *)
                string to prompt the user
    return:     user input converted to integer */
```

```c
int getInteger(const char *prompt)
{
    int myInt = 0;  // int to return

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

    return myInt;           // return integer
}
```

```
/*  HOMEWORK 2.2
    lowpower.c:
    Prompts the user for a VOLTAGE (float) & RESISTANCE (float).
    Calculates POWER (float):
        P = V^2 / R
    If POWER EXCEEDS MAX_POWER (0.25) W, print error & start over.
    Else print calculated POWER.
    Tom Grushka
    February 5, 2016    */

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

const float MAX_POWER = 0.25;   // Watts of MAX POWER allowed (0.25 in assignment
    spec)

// A few quotes from 2001: Space Odyssey to throw at the user for entering bad values
const char *QUOTES[6] = {
    "I'm sorry, Dave. I'm afraid I can't do that.",
    "I think you know what the problem is just as well as I do.",
    "This mission is too important for me to allow you to jeopardize it.",
    "I know that you and Frank were planning to fry me, and I'm afraid that's
something I cannot allow to happen.",
    "Daisy, Daisy, give me your answer do.",
    "Dave, this conversation can serve no purpose anymore. Goodbye."
};

// Declare input function
float getFloat(const char *prompt);

int main(int argc, char** argv)
{
    float voltage = 0.0;    // store input voltage
    float resistance = 0.0; // store input resistance
    float power = 0.0;      // store calculated power = voltage^2 / resistance
    int quote = 0;          // quote "counter"

    // Print introduction to user
    printf("Good afternoon, Dave. I am a HAL 9000 computer. I am afraid my power
needs adjusting.\n");

    /*  Loop indefinitely if power is a "bad value":
        less than zero or greater than MAX_POWER    */
    while (power <= 0 || power > MAX_POWER) {
        // Print instructions
        printf("Please specify a new voltage and resistance, Dave.\n");
        printf("I cannot handle more than %0.2f Watts of power, Dave.\n", MAX_POWER);
```

```c
        /*  Input loop: keep prompting user for voltage, then
             resistance, each until reasonable value entered.   */
        do {
            voltage = getFloat("Voltage: ");
        } while (voltage <= 0);
        do {
            resistance = getFloat("Resistance: ");
        } while (resistance <= 0);

        //  Calculate the power:    power = voltage ^2 / resistance
        power = (float)(pow(voltage, 2)) / resistance;

        //  Cannot exceed MAX_POWER!
        if (power > MAX_POWER)
        {
            printf("\a\a\aPower limit exceeded! %s\n\n", QUOTES[quote]);
            quote++;
            // Give the user 6 tries, then give up (exit with an error)
            if (quote == 6) exit(1);
        }
    }

    // Print positive feedback and result
    printf("\nVery good, Dave. That will be %0.2f Watts of power. Have a nice
day!\n", power);

    return 0;   // We shouldn't have an error
}

/*  getFloat
    argument:   prompt (const char *)
                string to prompt the user
    return:     user input converted to float   */
float getFloat(const char *prompt)
{
    float myFloat = 0;  // float to return

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

    return myFloat;         // return integer
}
```

```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   */
```

```c
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
}
```