Problem 1

Write a program that takes a value “n” as input and prints “Hello, World” n times.

From an analysis of your code, give a function representing the running time of your code. Give a tight asymptotic bound for that function.

```
void hello (int n)
{
    int i;
    for (i=0;i<n;i++) {
        printf is constant, c, and called n times
        printf("Hello World\n");
    }
}
```

A function representing time, f(n) = c * n.

f(n) is Θ (n)

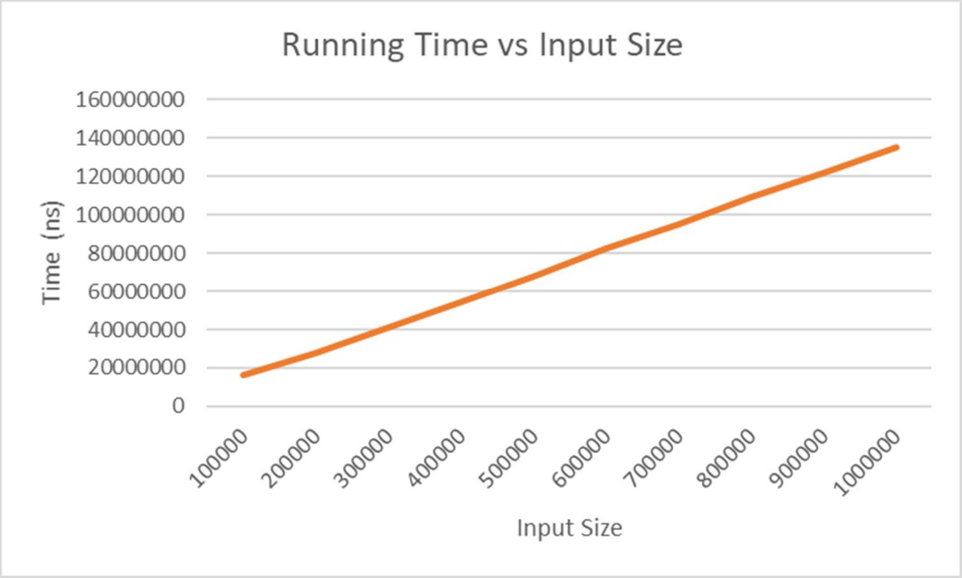
Run your code for various values of n and time it,

- *Create a chart showing the running times for various values of “n”,*
- *Create a graph of the running times vs various values of “n”. Use a linear scale on the axes.*
- *Describe how the running times support your analysis of the asymptotic running times.*

Here is the running time chart:

n	time (ns)
100000	15997200
200000	27428300
300000	41204600
400000	54185800
500000	67912600
600000	82214500
700000	94798100
800000	109273800
900000	121462300
1000000	135008200

Here is the running time graph. The axes are linear in scale so a linear relationship looks like a line!



The timing analysis supports my asymptotic analysis of Θ (n) because as the input size doubles (500,000 to 1,000,000) the running time essentially doubles (67.9ms to 135ms)

Give an estimation of how long it would take for n = 1 trillion

If 1,000,000 is 135ms, then 1,000,000,000,000 would be 1,000,000 times as large so the time would be 1,000,000 times as large or 135,000 seconds.

Here is the code in C: The highlighted code is the tested code. The rest of the code is the scaffolding code that is common for all problems:

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

void hello (int n)
{
    int i;
    for (i=0;i<n;i++) {
        printf("Hello World\n");
    }
}

unsigned long long get_time(){
    struct timespec time_val;
    unsigned long long val;

    clock_gettime(CLOCK_REALTIME, &time_val);
    val = time_val.tv_nsec + time_val.tv_sec*1000000000;
    return (val);
}

int main (int argc, char **argv)
{
    int min, max, step, count;
    int i,n;
    unsigned long long start_time, stop_time;

    min = atoi (argv[1]);
    max = atoi (argv[2]);
    step = atoi (argv[3]);

    for (n=min; n <= max; n+= step){
        start_time = get_time();
        hello(n);
        stop_time = get_time();
        printf("n= %d %lld\n",n,stop_time-start_time);
    }
}
```