```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
// Prototypes
double mag (double x, double y, double z);
double getDistanceFallen (double finalVelocity, double seconds);
bool closeTo (double tolerance, double target, double val);
int main () {
    bool fallen, falling;
    int dotI, exclamationI, time, startFall, stopFall;
    double x, y, z, acceleration, distanceFallen, fallTime;
    dotI = 0;
    exclamationI = 0;
    fallen = false;
    printf("OK, I'm now receiving data.\n");
    printf("I'm waiting");
    while (!fallen) {
        scanf("%d, %lf, %lf, %d, %d, %d, %d, %d, %d", &time, &x, &y, &z);
        acceleration = mag(x, y, z);
        // Prints the right number of dots after "I'm waiting"
        if (dotI > 20) {
           printf(".");
            fflush(stdout);
            dotI = 0;
        }
        // The Esplora is falling if the magnitude of it's acceleration is no longer close to 1
        if (!closeTo(0.25, 1, acceleration)) {
            printf("\n\tHelp me! I'm falling");
            startFall = time;
            falling = true;
            while (falling) {
                // Print exclamations marks while the Esplora is falling
                if (exclamationI > 12) {
                    printf("!");
                    fflush(stdout);
                    exclamationI = 0;
                }
                scanf("%d, %lf, %lf, %d, %d, %d, %d, %d, %d", &time, &x, &y, &z);
                acceleration = mag(x, y, z);
                // The Esplora is no longer falling when the magnitude of it's acceleration is
                close to 1
                if (closeTo(0.25, 1.0, acceleration)) {
                    falling = false;
                    stopFall = time;
                    fallen = true;
```

```
exclamationI++;
            }
        }
        dotI++;
    }
    fallTime = (stopFall - startFall) * 0.001; // Multiplying by 0.001 converts milliseconds to
    distanceFallen = getDistanceFallen(acceleration, fallTime);
    printf("\n\t\tOuch! I fell %.31f meters in %.101f seconds\n", distanceFallen, fallTime);
    return 0 ;
}
double mag (double x, double y, double z) {
    double magnitude = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
    return magnitude;
bool closeTo(double tolerance, double target, double val) {
    double lower, higher;
    lower = (target - tolerance);
    higher = (target + tolerance);
    if (val > lower && val < higher) {</pre>
        return true;
    } else {
        return false;
double getDistanceFallen (double acceleration, double seconds) {
    return (0.5 * (acceleration * 9.8) * pow(seconds, 2)); // Using the equation d = .5*a*t^2
}
```

```
cmbrooks@CO2018-16 /cygdrive/x/Documents/github/cpre185/labs/lab5/src
s ./../utilities/explore.exe -p COM12 -t -a -b | ./../bin/feat2
LEFT
BACK
UP
LEFT
FRONT
UP
FRONT
UP
FRONT
Cmbrooks@CO2018-16 /cygdrive/x/Documents/github/cpre185/labs/lab5/src
s |

cmbrooks@CO2018-16 /cygdrive/x/Documents/github/cpre185/labs/lab5/src
```