Department of Electrical and Computer Engineering
ECSE 202 – Introduction to Software Development
Assignment 5
Introduction to C Programming

Problem Description

This assignment introduces the concept of a command line program using the “C” programming language. In contrast to Java, the “C” programming environent is somewhat primitive (despite the fact that you can still develop code in Eclipse).

In this assignment you will port (translate) the code you wrote for Assignment 1 from java to “C”. There is no graphics support, so the program will simply print the values generated by the program (which can be compared to the output from Assignment 1).

Differences between “C” and Java (relative to Assignment 1)

1. There is no import statement. The closest analog in “C” is the include statement which literally inserts the specified file before compilation. For the purposes of this course, your program should include the following:

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

   When using Eclipse, these are added automatically.

2. Constants in “C” are not variables. The public static final expression in Java defined a static read-only variable that is used as a program constant. In “C” one uses a define statement to define a quantity or expression which is subsequently inserted in the source code by the “C” pre-processor. For example,

   ```
   public static final int NumBalls = 100;
   ```

   would be replaced in “C” by

   ```
   #define NumBalls 100
   ```

3. Input/Output in “C” is best explained by example. Consider the following Java code to read a parameter from the user:

   double Vo = readDouble ("Enter the initial velocity of the ball in meters/second [0,100]: ");

   In “C” this would be implemented as

   ```
   printf(("Enter the initial velocity of the ball in meters/second [0,100]: ");
   ```

```
scanf("%lf",&Vo);
```

Furthermore, Vo would have to be declared beforehand. For now treat this as a pattern which will be explained in subsequent lectures. Make sure you precede the variable name with an ampersand, &, as shown in the example.

Unlike the println method in Java, printf uses substitution in place of concatenation. For example, suppose you wanted to print the values of a, b, and c. Here is how you would do this using printf:

```
int a, c;
double b;

printf("The value of a is %d, the value of b is %f, and the value of c is %d\n",a,b,c);
```

Whenever printf encounters %, it interprets the next character as a type specifier for the next variable to be printed. In the above example, the first %d matches a, %f matches b, and the last %d matches c. In fact, the token string can actually embed a more detailed format specification – which will be covered in lectures. For now, here is a useful one for this assignment:

```
printf("This double value, %.2f, is printed with 2 decimal places\n",3.14159);
```

A couple of observations: i) the integer value after %. Is the number of decimal places, ii) unlike println you need to explicitly add \n to skip to a new line.

4. There is no boolean data type, integer values are used instead. It is often convenient to do the following:

```
#define false 0
#define true !false

while (true) {
}
```

Integer values are used in place of booleans with false corresponding to a zero value, and true **any** non-zero value.

Running your Program

This will be covered in the tutorials, but after you build your program you will need to run it. It is possible to do so inside Eclipse, but difficult for user input. The preferred method is to run your program inside a Command Line Interpreter:

1. Windows:
Click the Windows icon at the lower left hand corner of the screen. This brings up a panel with a search window at the bottom. Type CMD to launch the Windows CLI.

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

We'll assume that you wrote a version of the Hello World program using the Eclipse IDE (Integrated Development Environment). If you are new to programming, this is probably your safest bet. Otherwise you can use whatever tools suit your needs. In any case, to run your program from the CLI, you first need to know *where* your program resides in your computer's file hierarchy (there is a way around this, but for now let's assume not). If you used eclipse, then each project (all the files comprising your computer program) is usually located in C:\Users\<your user name>\workspace\<project name>. Notice that when you start the CLI on Windows, it defaults to C:\Windows\system32. In order to access our program we will have to change directories using the `cd` (change directory) command as follows:

```
C:\Windows\system32> cd c:\Users\ferrie\workspace

C:\Users\ferrie\workspace>dir
 Volume in drive C has no label.
 Volume Serial Number is 4841-8823

 Directory of c:\Users\ferrie\workspace

2017-06-28  10:19 AM    <DIR>          .
2017-06-28  10:19 AM    <DIR>          ..
2016-07-07  10:40 AM    <DIR>          .metadata
2016-09-12  04:59 PM    <DIR>          argDemo
2016-09-12  11:44 AM    <DIR>          classDemo
2016-07-07  10:41 AM    <DIR>          Hello-C
2016-07-07  11:07 AM    <DIR>          HelloWorld
2017-02-28  11:16 PM    <DIR>          JCalcGUI
2017-06-28  10:19 AM    <DIR>          myHello
               0 File(s)              0 bytes
               9 Dir(s)  112,799,371,264 bytes free

C:\Users\ferrie\workspace>
```

The default behavior of the windows CLI is to echo the current location before the command prompt. To list the contents of the workspace directory, I followed with a `dir` (directory) command. This lists all active projects in my directory. The "C" program I want to run is in project myHello, so I have to change directories again.

C:\Users\ferrie\workspace>cd myHello

```
C:\Users\ferrie\workspace\myHello>dir
 Volume in drive C has no label.
 Volume Serial Number is 4841-8823

 Directory of c:\Users\ferrie\workspace\myHello

2017-06-28  10:19 AM    <DIR>          .
2017-06-28  10:19 AM    <DIR>          ..
2017-06-28  10:19 AM            11,468 .cproject
2017-06-28  10:19 AM               785 .project
2017-06-28  10:19 AM    <DIR>          .settings
2017-06-28  10:19 AM    <DIR>          Debug
2017-06-28  10:19 AM    <DIR>          src
               2 File(s)         12,253 bytes
               5 Dir(s)  112,799,248,384 bytes free

C:\Users\ferrie\workspace\myHello>
```

Again, I used the `dir` command to list the contents of the `myHello` project folder. The actual program (executable) lives in the Debug directory, so we need to change directories one more time.

```
C:\Users\ferrie\workspace\myHello>cd Debug

C:\Users\ferrie\workspace\myHello\Debug>dir
 Volume in drive C has no label.
 Volume Serial Number is 4841-8823

 Directory of c:\Users\ferrie\workspace\myHello\Debug

2017-06-28  10:19 AM    <DIR>          .
2017-06-28  10:19 AM    <DIR>          ..
2017-06-28  10:19 AM            82,300 myHello.exe
2017-06-28  10:19 AM    <DIR>          src
               1 File(s)         82,300 bytes
               3 Dir(s)  112,799,326,208 bytes free
```

The file containing the program is the one with the `.exe` extension, i.e., `myHello.exe`.

To run the program, simply type the name (no need to include the extension):

```
C:\Users\ferrie\workspace\myHello\Debug>myHello
!!!Hello World!!!

C:\Users\ferrie\workspace\myHello\Debug>
```

Of course we could have simply cut to the chase and run the program directly by specifying the complete path:

```
C:\Windows\system32>c:\Users\ferrie\workspace\myHello\Debug\myHe
llo
!!!Hello World!!!

C:\Windows\system32>
```

The procedure is similar for Mac or Linux users. In the OS X environment, run the Terminal or XQuartz applications. For Linux, XTerm is the default application. In both cases, the CLI (or shell in Linux parlance) will start off in your home directory, so all you need to do is use the `cd` command to change to the appropriate workspace location, e.g.,

```
ferrie@lizard{myHello}: cd ~/Documents/workspace/myHello/Debug
ferrie@lizard{Debug}: ls
makefile        myHello         objects.mk      sources.mk
src
ferrie@lizard{Debug}:
```

Note a couple of subtle changes. First, the workspace directory is usually placed under the Documents folder in the Mac environment and under the home directory in the Linux environment. Instead of using the `dir` command, the `ls` (list) command is used.

Admittedly there are a lot of details here that can be somewhat overwhelming. The trick is to take things one step at a time and make use of online resources (Google is your friend) when you hit a wall. The first tutorial will help you to set things up on your own computer, so that your introduction to software development can be as painless as possible.

Instructions

1. Starting with Bounce.java from Assignment 1, translate this code into a “C” source file (Bounce.c) that compiles into a program that can be run from the command line as shown in the example below:

```
ferrie@Lizard{Debug}: Bounce
Enter the initial velocity of the ball in meters/second [0,100]: 40
Enter the launch angle in degrees [0,90]: 85
Enter energy loss parameter [0,1]: .4
Enter the radius of the ball in meters [0.1,5.0]: 1.0
t: 0.00  X: 5.00  Y: 1.00  Vx: -50.00  Vy: 0.00
t: 0.10  X: 5.35  Y: 4.93  Vx: 3.48  Vy: 39.32
t: 0.20  X: 5.70  Y: 8.76  Vx: 3.48  Vy: 38.26
t: 0.30  X: 6.04  Y: 12.48  Vx: 3.47  Vy: 37.20
t: 0.40  X: 6.39  Y: 16.09  Vx: 3.46  Vy: 36.15
t: 0.50  X: 6.73  Y: 19.60  Vx: 3.45  Vy: 35.10
t: 0.60  X: 7.08  Y: 23.01  Vx: 3.45  Vy: 34.05
t: 0.70  X: 7.42  Y: 26.31  Vx: 3.44  Vy: 33.00
t: 0.80  X: 7.77  Y: 29.50  Vx: 3.43  Vy: 31.96
t: 0.90  X: 8.11  Y: 32.60  Vx: 3.43  Vy: 30.91
t: 1.00  X: 8.45  Y: 35.58  Vx: 3.42  Vy: 29.87
t: 1.10  X: 8.79  Y: 38.47  Vx: 3.41  Vy: 28.83
t: 1.20  X: 9.13  Y: 41.25  Vx: 3.41  Vy: 27.80
t: 1.30  X: 9.47  Y: 43.92  Vx: 3.40  Vy: 26.76
t: 1.40  X: 9.81  Y: 46.49  Vx: 3.39  Vy: 25.73
t: 1.50  X: 10.15  Y: 48.96  Vx: 3.39  Vy: 24.70
t: 1.60  X: 10.49  Y: 51.33  Vx: 3.38  Vy: 23.67
t: 1.70  X: 10.83  Y: 53.60  Vx: 3.37  Vy: 22.64
t: 1.80  X: 11.16  Y: 55.76  Vx: 3.37  Vy: 21.62
```

2. Run your program and save your output to file Bounce.txt.

To Hand In

1. The source file, Bounce.c
2. The program output, Bounce.txt

This is a relatively short assignment with a 1-week deadline on November 18th at 5:00 pm.

About Coding Assignments

We encourage students to work together and exchange ideas. However, when it comes to finally sitting down to write your code, this must be done *independently*. Detecting software plagiarism is pretty much automated these days with systems such as MOSS.

https://www.quora.com/How-does-MOSS-Measure-Of-Software-Similarity-Stanford-detect-plagiarism

Please make sure your work is your own. If you are having trouble, the Faculty provides a free tutoring service to help you along. You can also contact the course instructor or the tutor during office hours. There are also numerous online resources – Google is your friend. The point isn't simply to get the assignment out of the way, but to actually learn something in doing.

fpf/Nov 10, 2019

Appendix - The Structure of a "C" Program (revisited)

The "C" programming language was written originally for a UNIX (predecessor of Linux) environment where interactive programs are run by a shell (CLI) program such as `sh`. A consequence of this is that a "C" program has the form of a *function*. Notice the similarity to Java.

```
int main (int argc, char *argv[])
{
  printf("Hello World!\n");
}
```

You can literally take this code, build the code (it will generate warnings), and run it. Let's look at things in more detail. This function, called `main`, takes 2 arguments. The first, `argc`, is an integer variable, and the second, `argv` is an array of character strings. We will go into more detail as to why this is so in future lectures. These variables are filled in by the CLI when the program is run from the command line. The function itself can return a value (it doesn't return anything here), which is enables programs to be combined together in scripts (another topic). For this assignment we need to figure out how to pass arguments from the command line to the program – time for another example.

```
1     #include <stdio.h>
2
3     int main(int argc, char *argv[]) {
4       int a, b;
5       if (argc != 3) {
6         printf("wrong number of arguments\n");
7         return(0);
8       }
9       sscanf(argv[1],"%d",&a);
10      sscanf(argv[2],"%d",&b);
11      printf("%d + %d = %d\n", a, b, a+b);
12    }
```

Let's assume that I created a file called plus.c containing the above code (in a project called plus within Eclipse). What will it do? Let's evaluate this bit of code line by line:

1. Any time you use a function in a "C" program, you have to provide a definition that defines its parameters. These are usually contained in an include file. The notation used above references a system include file for the functions used in this program, i.e., `sscanf` and `printf`.
2. Blank lines can increase the readability of a program.
3. The CLI views each command line as a set of character strings separated by whitespace, i.e., characters such as spaces and tabs. For example if you entered

   ```
   ferrie@FastCat{ferrie}: plus 3 5
   ```

the command line arguments would consist of 3 character strings, `plus, 3 and 5` respectively. In this case variable `argc` would have a value of 3. The structure of the `argv` variable is a bit more complicated. As we will see later in these lectures, character strings are represented by what is called a *pointer*, a variable that holds the memory address (i.e. points to) of the first character in the string. This is not unlike Java where argv would be a reference to a block of memory on the heap. So `argv` corresponds to, in fact, an array of pointers, one for each character string in the command line. In this example, `argv[0]` corresponds to the character string `plus`, `argv[1]` to the character string for 3 and `argv[2]` to `5`. The details of these representations will be discussed in more detail in the subsequent lectures. For now all you need to know is how to gain access to the command line arguments.

4. Every piece of information that is explicitly represented in a computer program must have a corresponding variable with a data type that matches its usage. Since this program will compute the sum of two integers, they are explicitly represented by variables `a` and `b`.
5. Any program that interacts with user input needs to do at least some rudimentary checking to make sure that sufficient data has been received for the program to produce the expected output. In this case, since we're adding two integer values, we need to make sure that 2 arguments have been supplied. In this case `argc=3`; recall that `argc` returns the total number of tokens on the command line, including the program name.
6. If the number of arguments is wrong, we send a message to the user using the `printf` function. Without going into detail, `printf` displays anything between the double quotes literally, except for tokens indicated with the `%` character. In the example above, the first instance of `%d` means convert the value of the first matching variable, i.e. `a`, into a string of numbers representing a decimal number, and substitute into the expression. The next instance does the same for variable `b`, and the final one for the value of `a+b`.
7. This causes the program to terminate and return to the CLI.
8. Termination of the `if` clause.
9. Recall that `argv[1]` corresponds to the character string 3 (or whatever the user entered, e.g., 52764), that corresponds to a decimal number. Here the `sscanf` function is used to convert a string of characters, `argv[1]`, that represent a decimal number, `%d`, into the computer's internal binary representation, `b`. The ampersand character, &, passes the location of `b` in memory to `sscanf` so that the converted value can be returned directly.
10. Do the same for the second argument and return the value to variable `b`.
11. At this point we succeeded in reading two values from the command line and storing them as binary variables. Here we compute the sum, convert the result to a character string, and print out the result using the `printf` function as in Line 6. Notice that the third instance of `%d`, which corresponds to the sum, operates on an expression (a+b) instead of a single variable.
12. Close of program.

```
/**
 * ECSE 202 Programming Assignment 5 - Fall 2019
 *
 * Port the Java code from Assignment 1 to "C"
 *
 * Since there is no graphical display available, the ball
 * trajectory will be displayed as a table in text form.
 * Minimal changes are made to the original Java code
 *
 * Based on the examples shown in Roberts Chapter 2, Programming by Example,
 * write a simple Java program to simulate a ball undergoing projectile
 * motion.  The ball is launched from the lower left hand side of the
 * screen and follows the typical parabolic trajectory.  When the ball
 * hits the ground, we use conservation of energy to determine a new
 * initial velocity assuming a constant energy loss per bounce.  From
 * here we can repeat the simulation until a point is reached where the
 * ball stops bouncing.  The easiest way to do this is to write a class
 * (Bounce) that extends the GraphicsProgram class from the acm package
 * discussed in class.
 *
 * The program should read 4 arguments from the console:
 * Vo:      a double corresponding to the initial velocity of
 *          the ball in MKS units (meters/second).
 * theta:   a double representing the launch angle of the ball [0,90].
 * loss:    a double corresponding to energy loss on each bounce.
 * bSize:   a double corresponding to the radius of the sphere (needed to
 *          account for air resistance.
 *
 * Your simulation uses simple Newtonian mechanics
 * ref: https://farside.ph.utexas.edu/teaching/336k/Newtonhtml/node29.html
 *
 * Vt = m g / (4 Pi k bSize^2)  -->  terminal velocity
 * Assume m, k = 0.016 (assuming a 1kg mass at Vt of 100 mph).
 * Vt = g / (4 Pi 0.016 bSize^2)
 * x = Vo Vt cos(theta) (1 - exp(-g t / Vt)) / g
 * y = Vt / g (Vo sin(theta) + Vt) (1 - exp(- g t / Vt)) - Vt t
 *
 *  where g is the gravitational constant of 9.8 m/2^2
```

```
 *
 * @author Frank Ferrie
 *
 * @date   September 30, 2019, Ported November 9, 2019
 *
 *
 */

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

int main() {

// Parameters used in this program

// 1. Paramters defined in screen coordinates (pixels, acm coordinates)

    #define WIDTH = 600                  // n.b. screen coordinates
    #define HEIGHT = 600
    #define OFFSET = 200

// 2. Parameters defined in simulation coordinates (MKS, x-->range, y-->height)

    #define g 9.8                        // MKS
    #define k 0.0016                     // Vt constant
    #define Pi 3.1416
    #define Xinit 5.0                    // Initial ball location (X)
    #define TICK 0.1                     // Clock tick duration (sec)
    #define ETHR 0.01                    // If either Vx or Vy < ETHR STOP
    #define XMAX 100.0                   // Maximum value of X
    #define YMAX 100.0                   // Maximum value of Y
    #define PD 1;                        // Trace point diameter
    #define SCALE = HEIGHT/XMAX      // Pixels/meter

// Miscellaneous parameters

    #define false 0
    #define true  !false
```

```c
//  Variables for user input

    double Vo, theta, loss, bSize;          // Parameters read from user
    double Vt;


// User Input

    printf("Enter the initial velocity of the ball in meters/second [0,100]: ");
    scanf("%lf",&Vo);
    printf("Enter the launch angle in degrees [0,90]: ");
    scanf("%lf",&theta);
    printf("Enter energy loss parameter [0,1]: ");
    scanf("%lf",&loss);
    printf("Enter the radius of the ball in meters [0.1,5.0]: ");
    scanf("%lf",&bSize);

// Simulation Strategy:
//
// For the most part, the ball follows a parabolic trajectory defined by the expressions
// for x,y (defined above).  So given an initial set of parameters (Vo, theta, loss, bSize),
// substitute increasing time values into the expressions to determine the current value of
// (x,y).
//
// At some point the ball will hit the ground - Voy is negative, and y <= bSize.
// Rather than explicitly modeling collisions, we approximate by calculating the
// kinetic energy before impact, scaling it by a loss parameter (e.g. 0.4 means a
// loss of 40%), and then re-starting the trajectory with a new initial velocity.
//
// This process is allowed to repeat until either the vertical or horizontal
// component of kinetic energy falls below a threshold value ETHR.
//
// Simulation starts at starting point of the ball.
// Note that the the Y axis is flipped vertically!
// Coordinate conversion should be moved to a method rather than converting on
// the fly as done below, but since we might have not covered methods in detail
// yet, it is not done here.
```

```
//
// Note:  Position is defined by ball center as opposed to the upper left hand
//        corner of the bounding box.
//
// Simulate a parabolic trajectory
// Units are MKS with mass = 1.0 Kg
//
    Vt = g / (4*Pi*bSize*bSize*k);      // Terminal velocity
//
// Time is reset at the start of each interval.
// Y corresponds to the current Y position in the interval
// X corresponds to the current X position in the interval
// Xo corresponds to the absolute X position (it is summed over
// all intervals to provide a cumulative S displacement.

    int numBounces=0;                   // used to count bounces (for Part 1 of the question)
    double time = 0;                    // time (reset at each interval)
    double KEx=ETHR;                    // Kinetic energy in X and Y directions
    double KEy=ETHR;
    double X,Xo,Xlast,Vx,Y,Vy,Ylast;    // Position and velocity variables as defined above

    double Vox=Vo*cos(theta*Pi/180);    // Initial velocity components in X and Y
    double Voy=Vo*sin(theta*Pi/180);
    Xo=Xinit;                           // Initial X position
    Y=bSize;                            // Initial Y position
    Ylast=Y;                            // Y position at end of previous iteration (use this
                                        // to estimate Y velocity).
    Xlast=Xo;                           // Same for X.

// Simulation loop - compute position and velocity using Newtonian mechanics

    while(true) {

        X = Vox*Vt/g*(1-exp(-g*time/Vt));               // Update position
        Y = bSize + Vt/g*(Voy+Vt)*(1-exp(-g*time/Vt))-Vt*time;

        Vx = (X-Xlast)/TICK;                                // Estimate X and Y velocities
        Vy = (Y-Ylast)/TICK;
```

```
// If the TEST parameter is true, print the current simulation values
// at each time step.

        printf("t: %.2f  X: %.2f  Y: %.2f  Vx: %.2f  Vy: %.2f\n",time,Xo+X,Y,Vx,Vy);

        Xlast = X;                                          // For next iteration
        Ylast = Y;


// Check to see if we've hit the ground.  If yes, inject energy loss,
// force current value of Y to ball radius, restart Yi for next
// interation.

        if ((Vy<0)&&(Y<=bSize)) {
            printf("bounce\n");
            KEx = 0.5*Vx*Vx*(1-loss);                 // Kinetic energy in X direction after collision
            KEy = 0.5*Vy*Vy*(1-loss);                 // Kinetic energy in Y direction after collision
            Vox = sqrt(2*KEx);                        // Resulting horizontal velocity
            Voy = sqrt(2*KEy);                        // Resulting vertical velocity
            time=0;                                   // Reset current interval time
            Y=bSize;                                  // Physically limit ball penetration
            Xo+=X;                                    // Add to the cumulative X displacement
            X=0;                                      // Reset X to zero for start of next interval
            Xlast=X;                                  // Reset last X and Y positions
            Ylast=Y;
            numBounces++;                             // Count bounces - optional
        }

// Termination condition - energy depletes to 0
        if ((KEy < ETHR) | (KEx < ETHR)) break;

// Update ball position on screen
// (note that the current position in sumulation coordinates is
// (Xo+x,Y).


//      pause(TICK*500);                    // Let display catch up
```

```
        time+=TICK;                         // Update time
    }
}
```

```
ferrie@Lizard{Debug}: Bounce
Enter the initial velocity of the ball in meters/second [0,100]: 40
Enter the launch angle in degrees [0,90]: 85
Enter energy loss parameter [0,1]: .4
Enter the radius of the ball in meters [0.1,5.0]: 1
t: 0.00  X: 5.00  Y: 1.00  Vx: -50.00  Vy: 0.00
t: 0.10  X: 5.35  Y: 4.93  Vx: 3.48  Vy: 39.32
t: 0.20  X: 5.70  Y: 8.76  Vx: 3.48  Vy: 38.26
t: 0.30  X: 6.04  Y: 12.48  Vx: 3.47  Vy: 37.20
t: 0.40  X: 6.39  Y: 16.09  Vx: 3.46  Vy: 36.15
t: 0.50  X: 6.73  Y: 19.60  Vx: 3.45  Vy: 35.10
t: 0.60  X: 7.08  Y: 23.01  Vx: 3.45  Vy: 34.05
t: 0.70  X: 7.42  Y: 26.31  Vx: 3.44  Vy: 33.00
t: 0.80  X: 7.77  Y: 29.50  Vx: 3.43  Vy: 31.96
t: 0.90  X: 8.11  Y: 32.60  Vx: 3.43  Vy: 30.91
t: 1.00  X: 8.45  Y: 35.58  Vx: 3.42  Vy: 29.87
t: 1.10  X: 8.79  Y: 38.47  Vx: 3.41  Vy: 28.83
t: 1.20  X: 9.13  Y: 41.25  Vx: 3.41  Vy: 27.80
t: 1.30  X: 9.47  Y: 43.92  Vx: 3.40  Vy: 26.76
t: 1.40  X: 9.81  Y: 46.49  Vx: 3.39  Vy: 25.73
t: 1.50  X: 10.15  Y: 48.96  Vx: 3.39  Vy: 24.70
t: 1.60  X: 10.49  Y: 51.33  Vx: 3.38  Vy: 23.67
t: 1.70  X: 10.83  Y: 53.60  Vx: 3.37  Vy: 22.64
t: 1.80  X: 11.16  Y: 55.76  Vx: 3.37  Vy: 21.62
t: 1.90  X: 11.50  Y: 57.82  Vx: 3.36  Vy: 20.60
t: 2.00  X: 11.83  Y: 59.77  Vx: 3.35  Vy: 19.58
t: 2.10  X: 12.17  Y: 61.63  Vx: 3.35  Vy: 18.56
t: 2.20  X: 12.50  Y: 63.38  Vx: 3.34  Vy: 17.54
t: 2.30  X: 12.84  Y: 65.04  Vx: 3.33  Vy: 16.53
t: 2.40  X: 13.17  Y: 66.59  Vx: 3.33  Vy: 15.51
t: 2.50  X: 13.50  Y: 68.04  Vx: 3.32  Vy: 14.50
t: 2.60  X: 13.83  Y: 69.39  Vx: 3.31  Vy: 13.50
t: 2.70  X: 14.16  Y: 70.64  Vx: 3.31  Vy: 12.49
t: 2.80  X: 14.49  Y: 71.79  Vx: 3.30  Vy: 11.49
t: 2.90  X: 14.82  Y: 72.83  Vx: 3.29  Vy: 10.48
t: 3.00  X: 15.15  Y: 73.78  Vx: 3.29  Vy: 9.48
t: 3.10  X: 15.48  Y: 74.63  Vx: 3.28  Vy: 8.49
t: 3.20  X: 15.80  Y: 75.38  Vx: 3.27  Vy: 7.49
t: 3.30  X: 16.13  Y: 76.03  Vx: 3.27  Vy: 6.50
t: 3.40  X: 16.46  Y: 76.58  Vx: 3.26  Vy: 5.50
t: 3.50  X: 16.78  Y: 77.03  Vx: 3.25  Vy: 4.51
t: 3.60  X: 17.11  Y: 77.38  Vx: 3.25  Vy: 3.53
t: 3.70  X: 17.43  Y: 77.64  Vx: 3.24  Vy: 2.54
t: 3.80  X: 17.75  Y: 77.79  Vx: 3.23  Vy: 1.56
t: 3.90  X: 18.08  Y: 77.85  Vx: 3.23  Vy: 0.57
t: 4.00  X: 18.40  Y: 77.81  Vx: 3.22  Vy: -0.41
t: 4.10  X: 18.72  Y: 77.67  Vx: 3.21  Vy: -1.39
t: 4.20  X: 19.04  Y: 77.44  Vx: 3.21  Vy: -2.36
t: 4.30  X: 19.36  Y: 77.10  Vx: 3.20  Vy: -3.34
t: 4.40  X: 19.68  Y: 76.67  Vx: 3.19  Vy: -4.31
t: 4.50  X: 20.00  Y: 76.14  Vx: 3.19  Vy: -5.28
t: 4.60  X: 20.32  Y: 75.52  Vx: 3.18  Vy: -6.25
t: 4.70  X: 20.63  Y: 74.80  Vx: 3.17  Vy: -7.21
t: 4.80  X: 20.95  Y: 73.98  Vx: 3.17  Vy: -8.18
t: 4.90  X: 21.27  Y: 73.07  Vx: 3.16  Vy: -9.14
t: 5.00  X: 21.58  Y: 72.06  Vx: 3.16  Vy: -10.10
t: 5.10  X: 21.90  Y: 70.95  Vx: 3.15  Vy: -11.06
t: 5.20  X: 22.21  Y: 69.75  Vx: 3.14  Vy: -12.02
t: 5.30  X: 22.53  Y: 68.45  Vx: 3.14  Vy: -12.97
t: 5.40  X: 22.84  Y: 67.06  Vx: 3.13  Vy: -13.92
t: 5.50  X: 23.15  Y: 65.57  Vx: 3.12  Vy: -14.88
t: 5.60  X: 23.46  Y: 63.99  Vx: 3.12  Vy: -15.82
t: 5.70  X: 23.77  Y: 62.31  Vx: 3.11  Vy: -16.77
t: 5.80  X: 24.08  Y: 60.54  Vx: 3.11  Vy: -17.72
t: 5.90  X: 24.39  Y: 58.67  Vx: 3.10  Vy: -18.66
t: 6.00  X: 24.70  Y: 56.71  Vx: 3.09  Vy: -19.60
t: 6.10  X: 25.01  Y: 54.66  Vx: 3.09  Vy: -20.54
t: 6.20  X: 25.32  Y: 52.51  Vx: 3.08  Vy: -21.48
```

```
t: 6.30  X: 25.63  Y: 50.27  Vx: 3.07  Vy: -22.42
t: 6.40  X: 25.94  Y: 47.93  Vx: 3.07  Vy: -23.35
t: 6.50  X: 26.24  Y: 45.51  Vx: 3.06  Vy: -24.28
t: 6.60  X: 26.55  Y: 42.98  Vx: 3.06  Vy: -25.21
t: 6.70  X: 26.85  Y: 40.37  Vx: 3.05  Vy: -26.14
t: 6.80  X: 27.16  Y: 37.66  Vx: 3.04  Vy: -27.07
t: 6.90  X: 27.46  Y: 34.87  Vx: 3.04  Vy: -27.99
t: 7.00  X: 27.76  Y: 31.97  Vx: 3.03  Vy: -28.91
t: 7.10  X: 28.07  Y: 28.99  Vx: 3.03  Vy: -29.84
t: 7.20  X: 28.37  Y: 25.91  Vx: 3.02  Vy: -30.75
t: 7.30  X: 28.67  Y: 22.75  Vx: 3.01  Vy: -31.67
t: 7.40  X: 28.97  Y: 19.49  Vx: 3.01  Vy: -32.59
t: 7.50  X: 29.27  Y: 16.14  Vx: 3.00  Vy: -33.50
t: 7.60  X: 29.57  Y: 12.70  Vx: 3.00  Vy: -34.41
t: 7.70  X: 29.87  Y: 9.17  Vx: 2.99  Vy: -35.32
t: 7.80  X: 30.17  Y: 5.54  Vx: 2.98  Vy: -36.23
t: 7.90  X: 30.46  Y: 1.83  Vx: 2.98  Vy: -37.14
t: 8.00  X: 30.76  Y: -1.98  Vx: 2.97  Vy: -38.04
bounce
t: 0.10  X: 30.99  Y: 3.89  Vx: 2.30  Vy: 28.95
t: 0.20  X: 31.22  Y: 6.69  Vx: 2.29  Vy: 27.91
t: 0.30  X: 31.45  Y: 9.37  Vx: 2.29  Vy: 26.87
t: 0.40  X: 31.68  Y: 11.96  Vx: 2.29  Vy: 25.84
t: 0.50  X: 31.91  Y: 14.44  Vx: 2.28  Vy: 24.81
t: 0.60  X: 32.13  Y: 16.82  Vx: 2.28  Vy: 23.78
t: 0.70  X: 32.36  Y: 19.09  Vx: 2.27  Vy: 22.76
t: 0.80  X: 32.59  Y: 21.27  Vx: 2.27  Vy: 21.73
t: 0.90  X: 32.81  Y: 23.34  Vx: 2.26  Vy: 20.71
t: 1.00  X: 33.04  Y: 25.30  Vx: 2.26  Vy: 19.69
t: 1.10  X: 33.27  Y: 27.17  Vx: 2.25  Vy: 18.67
t: 1.20  X: 33.49  Y: 28.94  Vx: 2.25  Vy: 17.65
t: 1.30  X: 33.71  Y: 30.60  Vx: 2.24  Vy: 16.64
t: 1.40  X: 33.94  Y: 32.16  Vx: 2.24  Vy: 15.63
t: 1.50  X: 34.16  Y: 33.62  Vx: 2.24  Vy: 14.61
t: 1.60  X: 34.39  Y: 34.99  Vx: 2.23  Vy: 13.61
t: 1.70  X: 34.61  Y: 36.25  Vx: 2.23  Vy: 12.60
t: 1.80  X: 34.83  Y: 37.40  Vx: 2.22  Vy: 11.60
t: 1.90  X: 35.05  Y: 38.46  Vx: 2.22  Vy: 10.59
t: 2.00  X: 35.27  Y: 39.42  Vx: 2.21  Vy: 9.59
t: 2.10  X: 35.49  Y: 40.28  Vx: 2.21  Vy: 8.59
t: 2.20  X: 35.71  Y: 41.04  Vx: 2.20  Vy: 7.60
t: 2.30  X: 35.93  Y: 41.70  Vx: 2.20  Vy: 6.60
t: 2.40  X: 36.15  Y: 42.26  Vx: 2.20  Vy: 5.61
t: 2.50  X: 36.37  Y: 42.73  Vx: 2.19  Vy: 4.62
t: 2.60  X: 36.59  Y: 43.09  Vx: 2.19  Vy: 3.63
t: 2.70  X: 36.81  Y: 43.35  Vx: 2.18  Vy: 2.65
t: 2.80  X: 37.03  Y: 43.52  Vx: 2.18  Vy: 1.66
t: 2.90  X: 37.24  Y: 43.59  Vx: 2.17  Vy: 0.68
t: 3.00  X: 37.46  Y: 43.56  Vx: 2.17  Vy: -0.30
t: 3.10  X: 37.68  Y: 43.43  Vx: 2.16  Vy: -1.28
t: 3.20  X: 37.89  Y: 43.21  Vx: 2.16  Vy: -2.25
t: 3.30  X: 38.11  Y: 42.88  Vx: 2.16  Vy: -3.23
t: 3.40  X: 38.32  Y: 42.46  Vx: 2.15  Vy: -4.20
t: 3.50  X: 38.54  Y: 41.95  Vx: 2.15  Vy: -5.17
t: 3.60  X: 38.75  Y: 41.33  Vx: 2.14  Vy: -6.14
t: 3.70  X: 38.97  Y: 40.62  Vx: 2.14  Vy: -7.11
t: 3.80  X: 39.18  Y: 39.81  Vx: 2.13  Vy: -8.07
t: 3.90  X: 39.39  Y: 38.91  Vx: 2.13  Vy: -9.04
t: 4.00  X: 39.61  Y: 37.91  Vx: 2.13  Vy: -10.00
t: 4.10  X: 39.82  Y: 36.81  Vx: 2.12  Vy: -10.96
t: 4.20  X: 40.03  Y: 35.62  Vx: 2.12  Vy: -11.91
t: 4.30  X: 40.24  Y: 34.34  Vx: 2.11  Vy: -12.87
t: 4.40  X: 40.45  Y: 32.95  Vx: 2.11  Vy: -13.82
t: 4.50  X: 40.66  Y: 31.48  Vx: 2.10  Vy: -14.77
t: 4.60  X: 40.87  Y: 29.91  Vx: 2.10  Vy: -15.72
t: 4.70  X: 41.08  Y: 28.24  Vx: 2.10  Vy: -16.67
t: 4.80  X: 41.29  Y: 26.48  Vx: 2.09  Vy: -17.61
t: 4.90  X: 41.50  Y: 24.62  Vx: 2.09  Vy: -18.56
```

```
t: 5.00  X: 41.71  Y: 22.67  Vx: 2.08  Vy: -19.50
t: 5.10  X: 41.92  Y: 20.63  Vx: 2.08  Vy: -20.44
t: 5.20  X: 42.12  Y: 18.49  Vx: 2.08  Vy: -21.38
t: 5.30  X: 42.33  Y: 16.26  Vx: 2.07  Vy: -22.31
t: 5.40  X: 42.54  Y: 13.93  Vx: 2.07  Vy: -23.25
t: 5.50  X: 42.74  Y: 11.52  Vx: 2.06  Vy: -24.18
t: 5.60  X: 42.95  Y: 9.00  Vx: 2.06  Vy: -25.11
t: 5.70  X: 43.16  Y: 6.40  Vx: 2.05  Vy: -26.04
t: 5.80  X: 43.36  Y: 3.70  Vx: 2.05  Vy: -26.97
t: 5.90  X: 43.57  Y: 0.92  Vx: 2.05  Vy: -27.89
bounce
t: 0.10  X: 43.72  Y: 3.11  Vx: 1.58  Vy: 21.09
t: 0.20  X: 43.88  Y: 5.12  Vx: 1.58  Vy: 20.07
t: 0.30  X: 44.04  Y: 7.02  Vx: 1.58  Vy: 19.05
t: 0.40  X: 44.20  Y: 8.82  Vx: 1.57  Vy: 18.03
t: 0.50  X: 44.35  Y: 10.53  Vx: 1.57  Vy: 17.02
t: 0.60  X: 44.51  Y: 12.13  Vx: 1.57  Vy: 16.01
t: 0.70  X: 44.67  Y: 13.63  Vx: 1.56  Vy: 14.99
t: 0.80  X: 44.82  Y: 15.03  Vx: 1.56  Vy: 13.99
t: 0.90  X: 44.98  Y: 16.32  Vx: 1.56  Vy: 12.98
t: 1.00  X: 45.13  Y: 17.52  Vx: 1.55  Vy: 11.97
t: 1.10  X: 45.29  Y: 18.62  Vx: 1.55  Vy: 10.97
t: 1.20  X: 45.44  Y: 19.61  Vx: 1.55  Vy: 9.97
t: 1.30  X: 45.60  Y: 20.51  Vx: 1.55  Vy: 8.97
t: 1.40  X: 45.75  Y: 21.31  Vx: 1.54  Vy: 7.97
t: 1.50  X: 45.91  Y: 22.01  Vx: 1.54  Vy: 6.98
t: 1.60  X: 46.06  Y: 22.61  Vx: 1.54  Vy: 5.99
t: 1.70  X: 46.21  Y: 23.10  Vx: 1.53  Vy: 4.99
t: 1.80  X: 46.37  Y: 23.51  Vx: 1.53  Vy: 4.01
t: 1.90  X: 46.52  Y: 23.81  Vx: 1.53  Vy: 3.02
t: 2.00  X: 46.67  Y: 24.01  Vx: 1.52  Vy: 2.03
t: 2.10  X: 46.82  Y: 24.12  Vx: 1.52  Vy: 1.05
t: 2.20  X: 46.98  Y: 24.12  Vx: 1.52  Vy: 0.07
t: 2.30  X: 47.13  Y: 24.03  Vx: 1.51  Vy: -0.91
t: 2.40  X: 47.28  Y: 23.84  Vx: 1.51  Vy: -1.89
t: 2.50  X: 47.43  Y: 23.56  Vx: 1.51  Vy: -2.86
t: 2.60  X: 47.58  Y: 23.17  Vx: 1.51  Vy: -3.84
t: 2.70  X: 47.73  Y: 22.69  Vx: 1.50  Vy: -4.81
t: 2.80  X: 47.88  Y: 22.11  Vx: 1.50  Vy: -5.78
t: 2.90  X: 48.03  Y: 21.44  Vx: 1.50  Vy: -6.74
t: 3.00  X: 48.18  Y: 20.67  Vx: 1.49  Vy: -7.71
t: 3.10  X: 48.33  Y: 19.80  Vx: 1.49  Vy: -8.67
t: 3.20  X: 48.48  Y: 18.84  Vx: 1.49  Vy: -9.63
t: 3.30  X: 48.63  Y: 17.78  Vx: 1.48  Vy: -10.59
t: 3.40  X: 48.77  Y: 16.62  Vx: 1.48  Vy: -11.55
t: 3.50  X: 48.92  Y: 15.37  Vx: 1.48  Vy: -12.51
t: 3.60  X: 49.07  Y: 14.03  Vx: 1.48  Vy: -13.46
t: 3.70  X: 49.22  Y: 12.58  Vx: 1.47  Vy: -14.41
t: 3.80  X: 49.36  Y: 11.05  Vx: 1.47  Vy: -15.36
t: 3.90  X: 49.51  Y: 9.42  Vx: 1.47  Vy: -16.31
t: 4.00  X: 49.66  Y: 7.69  Vx: 1.46  Vy: -17.26
t: 4.10  X: 49.80  Y: 5.87  Vx: 1.46  Vy: -18.20
t: 4.20  X: 49.95  Y: 3.96  Vx: 1.46  Vy: -19.15
t: 4.30  X: 50.09  Y: 1.95  Vx: 1.46  Vy: -20.09
t: 4.40  X: 50.24  Y: -0.15  Vx: 1.45  Vy: -21.02
bounce
t: 0.10  X: 50.35  Y: 2.58  Vx: 1.12  Vy: 15.78
t: 0.20  X: 50.46  Y: 4.05  Vx: 1.12  Vy: 14.77
t: 0.30  X: 50.58  Y: 5.43  Vx: 1.12  Vy: 13.76
t: 0.40  X: 50.69  Y: 6.71  Vx: 1.12  Vy: 12.75
t: 0.50  X: 50.80  Y: 7.88  Vx: 1.11  Vy: 11.75
t: 0.60  X: 50.91  Y: 8.96  Vx: 1.11  Vy: 10.75
t: 0.70  X: 51.02  Y: 9.93  Vx: 1.11  Vy: 9.75
t: 0.80  X: 51.13  Y: 10.80  Vx: 1.11  Vy: 8.75
t: 0.90  X: 51.24  Y: 11.58  Vx: 1.11  Vy: 7.75
t: 1.00  X: 51.35  Y: 12.26  Vx: 1.10  Vy: 6.76
t: 1.10  X: 51.46  Y: 12.83  Vx: 1.10  Vy: 5.76
t: 1.20  X: 51.57  Y: 13.31  Vx: 1.10  Vy: 4.77
```

```
t: 1.30  X: 51.68  Y: 13.69  Vx: 1.10  Vy: 3.78
t: 1.40  X: 51.79  Y: 13.97  Vx: 1.09  Vy: 2.80
t: 1.50  X: 51.90  Y: 14.15  Vx: 1.09  Vy: 1.81
t: 1.60  X: 52.01  Y: 14.23  Vx: 1.09  Vy: 0.83
t: 1.70  X: 52.12  Y: 14.22  Vx: 1.09  Vy: -0.15
t: 1.80  X: 52.23  Y: 14.10  Vx: 1.09  Vy: -1.13
t: 1.90  X: 52.34  Y: 13.89  Vx: 1.08  Vy: -2.11
t: 2.00  X: 52.44  Y: 13.58  Vx: 1.08  Vy: -3.08
t: 2.10  X: 52.55  Y: 13.18  Vx: 1.08  Vy: -4.05
t: 2.20  X: 52.66  Y: 12.68  Vx: 1.08  Vy: -5.02
t: 2.30  X: 52.77  Y: 12.08  Vx: 1.08  Vy: -5.99
t: 2.40  X: 52.87  Y: 11.38  Vx: 1.07  Vy: -6.96
t: 2.50  X: 52.98  Y: 10.59  Vx: 1.07  Vy: -7.93
t: 2.60  X: 53.09  Y: 9.70  Vx: 1.07  Vy: -8.89
t: 2.70  X: 53.20  Y: 8.72  Vx: 1.07  Vy: -9.85
t: 2.80  X: 53.30  Y: 7.63  Vx: 1.06  Vy: -10.81
t: 2.90  X: 53.41  Y: 6.46  Vx: 1.06  Vy: -11.77
t: 3.00  X: 53.51  Y: 5.19  Vx: 1.06  Vy: -12.72
t: 3.10  X: 53.62  Y: 3.82  Vx: 1.06  Vy: -13.68
t: 3.20  X: 53.73  Y: 2.36  Vx: 1.06  Vy: -14.63
t: 3.30  X: 53.83  Y: 0.80  Vx: 1.05  Vy: -15.58
bounce
t: 0.10  X: 53.91  Y: 2.16  Vx: 0.82  Vy: 11.56
t: 0.20  X: 53.99  Y: 3.21  Vx: 0.81  Vy: 10.56
t: 0.30  X: 54.07  Y: 4.17  Vx: 0.81  Vy: 9.56
t: 0.40  X: 54.16  Y: 5.02  Vx: 0.81  Vy: 8.56
t: 0.50  X: 54.24  Y: 5.78  Vx: 0.81  Vy: 7.57
t: 0.60  X: 54.32  Y: 6.44  Vx: 0.81  Vy: 6.57
t: 0.70  X: 54.40  Y: 7.00  Vx: 0.81  Vy: 5.58
t: 0.80  X: 54.48  Y: 7.46  Vx: 0.80  Vy: 4.59
t: 0.90  X: 54.56  Y: 7.82  Vx: 0.80  Vy: 3.60
t: 1.00  X: 54.64  Y: 8.08  Vx: 0.80  Vy: 2.62
t: 1.10  X: 54.72  Y: 8.24  Vx: 0.80  Vy: 1.63
t: 1.20  X: 54.80  Y: 8.31  Vx: 0.80  Vy: 0.65
t: 1.30  X: 54.88  Y: 8.27  Vx: 0.80  Vy: -0.33
t: 1.40  X: 54.96  Y: 8.14  Vx: 0.79  Vy: -1.31
t: 1.50  X: 55.04  Y: 7.91  Vx: 0.79  Vy: -2.29
t: 1.60  X: 55.12  Y: 7.59  Vx: 0.79  Vy: -3.26
t: 1.70  X: 55.19  Y: 7.16  Vx: 0.79  Vy: -4.23
t: 1.80  X: 55.27  Y: 6.64  Vx: 0.79  Vy: -5.20
t: 1.90  X: 55.35  Y: 6.03  Vx: 0.79  Vy: -6.17
t: 2.00  X: 55.43  Y: 5.31  Vx: 0.78  Vy: -7.14
t: 2.10  X: 55.51  Y: 4.50  Vx: 0.78  Vy: -8.10
t: 2.20  X: 55.59  Y: 3.60  Vx: 0.78  Vy: -9.07
t: 2.30  X: 55.66  Y: 2.59  Vx: 0.78  Vy: -10.03
t: 2.40  X: 55.74  Y: 1.49  Vx: 0.78  Vy: -10.99
t: 2.50  X: 55.82  Y: 0.30  Vx: 0.78  Vy: -11.94
bounce
t: 0.10  X: 55.88  Y: 1.88  Vx: 0.60  Vy: 8.75
t: 0.20  X: 55.94  Y: 2.65  Vx: 0.60  Vy: 7.76
t: 0.30  X: 56.00  Y: 3.33  Vx: 0.60  Vy: 6.76
t: 0.40  X: 56.06  Y: 3.90  Vx: 0.60  Vy: 5.77
t: 0.50  X: 56.12  Y: 4.38  Vx: 0.60  Vy: 4.78
t: 0.60  X: 56.18  Y: 4.76  Vx: 0.60  Vy: 3.79
t: 0.70  X: 56.24  Y: 5.04  Vx: 0.59  Vy: 2.80
t: 0.80  X: 56.30  Y: 5.22  Vx: 0.59  Vy: 1.82
t: 0.90  X: 56.36  Y: 5.31  Vx: 0.59  Vy: 0.83
t: 1.00  X: 56.42  Y: 5.29  Vx: 0.59  Vy: -0.15
t: 1.10  X: 56.48  Y: 5.18  Vx: 0.59  Vy: -1.12
t: 1.20  X: 56.53  Y: 4.97  Vx: 0.59  Vy: -2.10
t: 1.30  X: 56.59  Y: 4.66  Vx: 0.59  Vy: -3.08
t: 1.40  X: 56.65  Y: 4.26  Vx: 0.59  Vy: -4.05
t: 1.50  X: 56.71  Y: 3.75  Vx: 0.58  Vy: -5.02
t: 1.60  X: 56.77  Y: 3.16  Vx: 0.58  Vy: -5.99
t: 1.70  X: 56.83  Y: 2.46  Vx: 0.58  Vy: -6.96
t: 1.80  X: 56.88  Y: 1.67  Vx: 0.58  Vy: -7.92
t: 1.90  X: 56.94  Y: 0.78  Vx: 0.58  Vy: -8.88
bounce
```

```
t: 0.10  X: 56.99  Y: 1.64  Vx: 0.45  Vy: 6.38
t: 0.20  X: 57.03  Y: 2.18  Vx: 0.45  Vy: 5.39
t: 0.30  X: 57.08  Y: 2.62  Vx: 0.45  Vy: 4.40
t: 0.40  X: 57.12  Y: 2.96  Vx: 0.45  Vy: 3.42
t: 0.50  X: 57.17  Y: 3.20  Vx: 0.45  Vy: 2.43
t: 0.60  X: 57.21  Y: 3.35  Vx: 0.44  Vy: 1.45
t: 0.70  X: 57.25  Y: 3.39  Vx: 0.44  Vy: 0.46
t: 0.80  X: 57.30  Y: 3.34  Vx: 0.44  Vy: -0.52
t: 0.90  X: 57.34  Y: 3.19  Vx: 0.44  Vy: -1.49
t: 1.00  X: 57.39  Y: 2.95  Vx: 0.44  Vy: -2.47
t: 1.10  X: 57.43  Y: 2.60  Vx: 0.44  Vy: -3.44
t: 1.20  X: 57.47  Y: 2.16  Vx: 0.44  Vy: -4.42
t: 1.30  X: 57.52  Y: 1.62  Vx: 0.44  Vy: -5.39
t: 1.40  X: 57.56  Y: 0.99  Vx: 0.44  Vy: -6.35
bounce
t: 0.10  X: 57.60  Y: 1.44  Vx: 0.34  Vy: 4.43
t: 0.20  X: 57.63  Y: 1.79  Vx: 0.34  Vy: 3.44
t: 0.30  X: 57.66  Y: 2.03  Vx: 0.34  Vy: 2.45
t: 0.40  X: 57.70  Y: 2.18  Vx: 0.34  Vy: 1.47
t: 0.50  X: 57.73  Y: 2.23  Vx: 0.34  Vy: 0.49
t: 0.60  X: 57.76  Y: 2.18  Vx: 0.33  Vy: -0.49
t: 0.70  X: 57.80  Y: 2.03  Vx: 0.33  Vy: -1.47
t: 0.80  X: 57.83  Y: 1.79  Vx: 0.33  Vy: -2.45
t: 0.90  X: 57.86  Y: 1.45  Vx: 0.33  Vy: -3.42
t: 1.00  X: 57.90  Y: 1.01  Vx: 0.33  Vy: -4.39
t: 1.10  X: 57.93  Y: 0.47  Vx: 0.33  Vy: -5.36
bounce
t: 0.10  X: 57.96  Y: 1.37  Vx: 0.26  Vy: 3.66
t: 0.20  X: 57.98  Y: 1.63  Vx: 0.26  Vy: 2.67
t: 0.30  X: 58.01  Y: 1.80  Vx: 0.26  Vy: 1.69
t: 0.40  X: 58.03  Y: 1.87  Vx: 0.25  Vy: 0.71
t: 0.50  X: 58.06  Y: 1.85  Vx: 0.25  Vy: -0.27
t: 0.60  X: 58.08  Y: 1.72  Vx: 0.25  Vy: -1.25
t: 0.70  X: 58.11  Y: 1.50  Vx: 0.25  Vy: -2.23
t: 0.80  X: 58.13  Y: 1.18  Vx: 0.25  Vy: -3.20
t: 0.90  X: 58.16  Y: 0.76  Vx: 0.25  Vy: -4.18
bounce
t: 0.10  X: 58.18  Y: 1.27  Vx: 0.20  Vy: 2.74
t: 0.20  X: 58.20  Y: 1.45  Vx: 0.19  Vy: 1.76
t: 0.30  X: 58.22  Y: 1.53  Vx: 0.19  Vy: 0.77
t: 0.40  X: 58.24  Y: 1.51  Vx: 0.19  Vy: -0.21
t: 0.50  X: 58.26  Y: 1.39  Vx: 0.19  Vy: -1.18
t: 0.60  X: 58.28  Y: 1.17  Vx: 0.19  Vy: -2.16
t: 0.70  X: 58.30  Y: 0.86  Vx: 0.19  Vy: -3.14
bounce
t: 0.10  X: 58.31  Y: 1.19  Vx: 0.15  Vy: 1.94
t: 0.20  X: 58.33  Y: 1.29  Vx: 0.15  Vy: 0.95
t: 0.30  X: 58.34  Y: 1.29  Vx: 0.15  Vy: -0.03
t: 0.40  X: 58.36  Y: 1.19  Vx: 0.15  Vy: -1.01
t: 0.50  X: 58.37  Y: 0.99  Vx: 0.15  Vy: -1.98
bounce
ferrie@Lizard{Debug}:
```