CS19001/CS19002 PROGRAMMING AND DATA STRUCTURES LABORATORY
Supplementary Assignment
Date: 02–February–2015 (Not for submission)

Part 1

Let P and Q be two points in the two-dimensional plane. We say that P *dominates* Q if both the conditions $x(P) \geq x(Q)$ and $y(P) \geq y(Q)$ are satisfied. A point P in a collection is called a *champion* if P dominates all other points in the collection. Not all collections contain champions (for example, consider the three points (1,1), (3,3), (2,4)). But if a champion exists in a collection, it is unique. In this case, the champion has both the largest x-coordinate and the largest y-coordinate in the collection.

Write a program that enters a loop. Each iteration of the loop body first reads the x- and y-coordinates of a new point (call them x and y). Assume that x and y are integer values. If either x or y (or both) is/are negative, then the program terminates. Otherwise, the champion among all the points read so far is printed, provided that the champion exists.

Maintain the largest x-coordinate and the largest y-coordinate read so far. Suppose that just before reading the current point (x, y), a champion existed. If the new point (x, y) dominates the old champion, then the new point now becomes the current champion. If the new point is dominated by the old champion, then the old champion continues to remain the current champion. Otherwise, the current champion ceases to exist.

Handle the case when the old champion (just before reading (x, y)) did not exist.

Sample output

```
New point: (678,  41). Current champion: (678,  41)
New point: ( 49, 223). Current champion: None
New point: (836, 300). Current champion: (836, 300)
New point: ( 98, 170). Current champion: (836, 300)
New point: ( 76, 853). Current champion: None
New point: (895, 929). Current champion: (895, 929)
New point: (266, 403). Current champion: (895, 929)
New point: (770, 725). Current champion: (895, 929)
New point: (594, 951). Current champion: None
New point: ( 15, 137). Current champion: None
New point: (388, 982). Current champion: None
New point: ( -1,  -1). Quitting...
```

Part 2

Suppose that you drop an apple from a height of N meters. After t seconds, the apple covers a distance of $\frac{1}{2}gt^2$. Assume that N is a positive integer. Let t_i be the time at which the apple covers i meters, where $0 \leq i \leq N$. Print in a line, for each i in this range, the time t_i and the speed of the apple at that time.

In order to have a visual effect of the free fall, print the i-th line at time t_i. This indicates that between the printings of the i-th line and the $(i + 1)$-st line, your program should wait for $(t_{i+1} - t_i)$ seconds. You may use the following call:

```
usleep(nms);
```

This causes the program to sleep (do nothing) for `nms` microseconds (an integer value). You need to include the header file `<unistd.h>`. No extra compile-time flag is necessary (you need only the `-lm` flag for square roots). Ignore the time taken by the computations and the printing in the loop body.

Sample output

```
Enter height: 10
               Time   Distance     Speed
   +++  0.000000      0      0.000000
   +++  0.451524      1      4.429447
   +++  0.638551      2      6.264184
   +++  0.782062      3      7.672027
   +++  0.903047      4      8.858894
   +++  1.009638      5      9.904544
   +++  1.106003      6     10.849885
   +++  1.194619      7     11.719215
   +++  1.277102      8     12.528368
   +++  1.354571      9     13.288341
   +++  1.427843     10     14.007141
   !!! BANG !!!! BANG !!!! BANG !!!
```

Note: Do not use arrays in any of the parts.