Bungee Jump

Due Date: 23:59 14 April 2019 – **can't be late** **Earnings:** 9% of your final grade.
Purpose: Solve Ordinary differential equations in real-time to simulate a bungee jump.

NOTE: Plan to finish a few days early to avoid last minute hardware/software or other unexpected holdups, for which no allowance is given.
NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.

Algorithm: While the user wishes to continue, the application will ask the user to select whether to run the simulation one more time. The simulation uses Heun's method, as explained in lectures, to solve a differential equation for acceleration under gravity, with controllable wind drag and with an additional elastic restoring term B so as to simulate falling off the top of a 200 metres high tower with a 10 metre length bungee cord tied to your ankles. The task is to reach the ground (height < 1.0 metres) at low speed (<1.0 metres/sec) so you can safely slip off the bungee and walk away. With no wind drag you'll probably hit the ground faster than 1metre/sec and die, but with too much drag you will not reach the ground al all but instead oscillate until you come to rest suspended 25m above the ground. Wind drag is adjustable from the keyboard 'w' increases it, 'e' decreases it. You can use _kbhit() to detect a key press and then _getch() to discover which key.
There is a "bare-bones" console executable, bungee.exe, on BB that that runs the simulation as a single shot – it does not offer a repeat. Your version should work the same, displaying the same information, but allow repeats.

The differential equations you are solving are:

1. the acceleration for falling under gravity with wind resistance and a bungee rope tied to your ankle:

   `dv/dt = = f(v,h) = g - (0.02*c)*(v + 0.004*v*v*v) - B(h)`

   where v is speed down, with values of the constants g = 9.8 and c(initially) = 0.0
   The terms on the right-hand side are:
   first term. The constant acceleration of gravity g pulling you down.
   second term – depends on v. The wind drag of the air slowing you down. It increases with speed v. It can be increased (press w to increase c up to a max of 20) or decreased (press e to decrease c down to a min of 0).
   third term – depends on h. The elastic restoring force B of the bungee cord pulling you up. The bungee is 10m long and until it is fully extended it has no effect. Once it is fully extended (>10m) it pulls you up with a force that gets larger the more it is extended. The force is equal to k*(bungee extension) where k is a constant equal to 0.06.

2. speed is the rate of change of height, which is solved by the Euler equation below.

You start the simulation at time t = 0 with height h = h_0 = 200m and speed down = 0. Thereafter the differential equation together with your input for c determine both v and h.

These differential equations can be solved in a stepwise iterative algorithm using Heun's method as discussed in class. Note that the acceleration 1. depends on the speed v through the drag, and the height h through the bungee extension, so you use the values for the last iteration i to calculate where you get to on the next iteration i+1. In the simplest Euler method the iterative equation is

$$v_{i+1} = v_i + accn_i(v_i,h_i)*time\_elapsed$$

where `time_elapsed` is the time in seconds that has elapsed since the last step as read from the system clock (see later).
Also the current height h_{i+1} in metres is calculated from the previous height iteratively using Euler as:

$$h_{i+1} = h_i - v_i*time\_elapsed$$

It makes sense to use Euler's method when you first write your code, but then implement Heun's

improvement before you hand it in.
When the simulation starts, you are 200 meters up on the tower. Then you fall to the ground under the control of the differential equations. The simulation continuously outputs the velocity and distance above the ground and reports whether the bungee is currently extended (so it's pulling you up). At the same time the overall duration of the jump is recorded.

Time. For each iteration you need the time in seconds that has elapsed since the last iteration, which is the step. There are several ways of doing this, but a simple one is to use the Windows function GetTickCount() that you can access by putting #include <windows.h> at the top of your file. This function returns the number of milliseconds that have elapsed since the system was started, up to 49.7 days. The precision of this timer is poor but you can choose not to execute the next iteration unless the step exceeds, say 20msec. However you computer also provides high performance timers that you may wish to investigate.

See the Marking Sheet for how you can lose marks, but you will lose at least 60% if:
1. it fails to build and run in Visual Studio 2017
2. It crashes in normal operation
3. it doesn't produce the example output.

Make sure you have submitted the correct file. If I cannot build it because the file is wrong or missing from the zip, even if it's an honest mistake, you get 0.

What to Do: Make an empty project in Visual Studio 2017, add a new source code file ass3.cpp to the project and write your code in it. Then on Brightspace in the Assignment Submission folder submit a zip file (**not** RAR, not 9zip, not 7 zip) containing your ass3.cpp. The name of the zip file **must** contain your name as a prefix so that I can identify it, for example using my name the file would be tyleraAss3CST8233.zip. It is also vital that you include the Information (as specified in the Submission Standard) as a file header in your source file so it can be identified as yours. Use comment lines in the file to include the header.
Because of Finals, it cannot be late. Don't send me the file as an email attachment – it will get 0.