Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
225157	0	0	0																			
270034	5	5	0	1	2	2	0	0														
293846	0	0	0																			
295323	0	0	0																			
345642	0	0	0																			
348843	0	0	0																			
349936	13.5	10	3.5	1	2	2	2	3		RK4: final linear combination is unweighted (missing factors 1,2,2,1)	1.5		2									
350475	10	10	0	1	2	2	2	3														
352091	0	0	0																			
353980	0	0	0																			
354439	8	9	0	1	2	1	2	3	-1	Empty readme. Only requirements checked - let us know if we missed something. R3: you seem to be using only f1 in the computation of the output state instead of 0.5* (f0+f1). xt is unused?											
355593	0	0	0																			
356026	12	10	2	1	2	2	2	3			2											
361749	0	0	0																			
369181	12	10	2	1	2	2	2	3			2											
372660	1	1	0	1	0	0	0	0														
387370	0	0	0																			
425575	0	0	0																			
425614	0	0	0																			
426419	0	0	0																			
427489	7	7	0	1	2	2	2	0														
428022	0	0	0																			
429487	0	0	0																			
430829	1	1	0	1	0	0	0	0														
457598	0	0	0																			
460297	0	0	0																			
464772	0	0	0																			
46477D	0	0	0																			
46596K	0	0	0																			
474199	13	9	4	1	2	2	1.5	2.5		R4+R5: you are using the variable current_state_ in the updates which contains the initial state of the step (but which is not modified by the integrators until the end of the step) - instead you should be using the State state, that is passed to evalF. This leads to the instability you described. The naming is a bit unfortunate here.	2				2							
474322	0	0	0																			
474458	10	10	0	1	2	2	2	3														
474898	4.5	4.5	0	1	1.5	2	0	0		R2: spring force function computes length only using x and y coords for a vec3f.												
475389	0	0	0																			
475813	0	0	0																			
475910	0	0	0																			
477329	12	10	2	1	2	2	2	3		R5: evalF is nearly unreadable and static to this particular configuration, but seems to work. RK4: cool use of the assignment operator mid arithmetic - I haven't though about it that way before	2											
477811	12	10	2	1	2	2	2	3			2											
478328	1	1	0	1	0	0	0	0														
478470	0	0	0																			
478687	0	0	0																			

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
479505	21	10	11	1	2	2	2	3		Tearing: maybe a bit odd to have a force threshold instead of a stretch limit (you could have a huge force for a tiny time interval but still have next to no movement due to the need of integration).	2	3	2		2		1	1				
479589	0	0	0																			
479741	9.5	9.5	0	1	2	2	2	2.5		R5: becomes unstable at smaller stepsizes than the example. At least the rest lengths are wrong (divide by x_-1 or y_-1, respectively), but also something else that I could not immediately notice.												
480086	0	0	0																			
480248	0	0	0																			
480714	12	10	2	1	2	2	2	3			2											
480798	0	0	0																			
481577	12	10	2	1	2	2	2	3			2											
493840	11	9	2	1	1	2	2	3			2											
506300	10	10	0	1	2	2	2	3														
508285	8	8	0	1	2	2	2	1														
51620U	0	0	0																			
524926	5	5	0	1	2	2	0	0														
525417	9	9	0	1	2	2	1.5	2.5		R4+R5: you are using the variable current_state_ in the updates which contains the initial state and is not updated - instead you should be using the State state, that is passed to evalF.												
525491	0	0	0																			
525941	0	0	0																			
526050	12	10	2	1	2	2	2	3			2											
526319	0	0	0																			
526775	0	0	0																			
527143	0	0	0																			
527389	0	0	0																			
528867	0	0	0																			
528883	9	7	2	1	2	2	2	0		R5: mere initial locations with no dynamics not quite enough for points.	2											
529196	0	0	0																			
529303	0	0	0																			
530185	0	0	0																			
530363	0	0	0																			
530619	0	0	0																			
530648	0	0	0																			
530868	15	10	5	1	2	2	2	3			2	3										
530981	13	10	3	1	2	2	2	3		Please ensure that the initial conditions match those of the example: yours were somewhat simpler. The code worked correctly, though, even with the example config.	2		1									
540094	0	0	0																			
540654	0	0	0																			
544566	9	5.5	3.5	1	2	1.5	1	0		R3: tiny indexing error: you are using f0[0] for every index i: should be f0[i], instead. R4: the rest length differs slightly (n particles -> n-1 intervals). There is also some sign mistake with the spring forces as the points are immediately drawn upwards. The logic is way too complex, as you noticed yourself, too! Valiant effort, nevertheless! Sprinkler not particularly exciting.	2	1.5										

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
549040	5	3	2	1	0	2	0	0		R2: spring force should have direction of the line connecting pos1 and pos2. Drag force has the wrong sign and is proportional to the location, not the velocity of the particle.	2											
549163	10	10	0	1	2	2	2	3		R3: looks like you are not clearing springs in reset, so the system feels a bit more rigid (-0p)												
55055P	12	10	2	1	2	2	2	3		Cool wind, very violent!	2											
552794	3	3	0	1	0	2	0	0														
552969	0	0	0																			
554598	12	10	2	1	2	2	2	3		Using std::transform, cool!	2											
563068	5	4	1	1	1	2	0	0		Your Euler-step actually takes two steps? R2: f[2] should not directly depend on the current force, but is rather the velocity-component of the current state. RK: You need no evaluate the forces after each RK-step and use that in the next part. The simple system diverges with your implementation.	1											
576149	0	0	0																			
585716	0	0	0																			
586333	0	0	0																			
586702	1	1	0	0.5	0.5	0	0	0		R1: eulerStep only increments the first element of the state, need a loop here!. R2: fDrag should take in the current velocity, not the acceleration and should oppose the movement (flip signs!).												
586980	7	7	0	1	2	2	2	0														
587316	0	0	0																			
587471	17.5	9.5	8	1	2	2	2	2.5		R5: clearing the springs_ vector at reset() helps for the instability! Cloth GPU sim with RK4.	2									6		
588289	10	10	0	1	2	2	2	3														
589291	0	0	0																			
589343	0	0	0																			
589848	3	3	0	1	2	0	0	0														
590921	0	0	0																			
591904	5	5	0	1	2	2	0	0														
591946	7	7	0	1	2	2	2	0														
592929	0	0	0																			
593274	10	10	0	1	2	2	2	3														
593847	0	0	0																			
594435	0	0	0																			
595926	12	10	2	1	2	2	2	3		R4: you are not clearing the springs vector at reset(), hence the different behavior (-0p)	2											
595997	0	0	0																			
596747	0	0	0																			
597429	10	10	0	1	2	2	2	3														
597623	11	10	1	1	2	2	2	3		RK4: k1 should use half step, k3 full step. Also, k4 not necessary, final step direction given by lin. comp of f's.	1											
597937	38	10	28	1	2	2	2	3		Cloth GPU sim with RK4. The implicit integrator convergence criterion is somewhat looser than in the example. Not entirely sure where the around 3x perf drop wrt reference comes from: would need to have a closer look at performance metrics.	2	3	2		2				8	6	5	RKF45 (3p), Impl. opt (2p)

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
602851	0	0	0																			
602893	12	10	2	1	2	2	2	3			2											
603096	12	10	2	1	2	2	2	3			2											
603245	10	10	0	1	2	2	2	3														
604095	34	10	24	1	2	2	2	3		Implicit integrator is really just an explicit integrator since without the Jacobian the solution of Ax=b is trivial and equals to h*F.	2	3	2	3	2		1		2	6	3	RKF45 (3p)
604273	13.5	10	3.5	1	2	2	2	3		RK4: there is really nothing extra compared to trapezoid implemented. Spray system is very simple and so is the rendering mode.	0	1	2					0.5				
606064	9.5	9.5	0	1	2	2	2	2.5		R5: after setting the initial position (change y and z coords of the system) to match the example and attaching the other corner (just find the 20th point!) everything works as intended.												
608949	0	0	0																			
609155	0	0	0																			
609249	0	0	0																			
612472	17	10	7	1	2	2	2	3			2		2								3	Dopri5 adaptive solver (3p)
612498	0	0	0																			
612870	17	10	7	1	2	2	2	3			2	3			2							
614577	0	0	0																			
614580	0	0	0																			
621308	14.5	10	4.5	1	2	2	2	3		Spray system quite simplistic and not that visually interesting. The effect of wind is barely visible after uncommenting the appropriate line: uniform but time-dependent force would probably have been more interesting. Tearing: could have e.g. just had an if-statement at evalF to decide whether to add forces or not given if the spring is broken or not.	2	1	1				0.5					
628835	9	7	2	1	2	2	2				2											
63036R	0	0	0																			
641922	0	0	0																			
646804	23	10	13	1	2	2	2	3			2		2		2		1			6		
647764	13.5	9.5	4	1	2	2	2	2.5		R5: You don't seem to be dividing by the mass.	2		2									
648530	0	0	0																			
648860	0	0	0																			
650191	9	9	0	1	2	2	1.5	2.5		R4, R5: evalF evaluated for wrong state (current_state_ instead of state)												
650227	0	0	0																			
650405	0	0	0																			
650560	6.5	6.5	0	1	2	2	1.5	0		R4: small indexing mistake: your lowermost spring affects the second and third-last points in the chain.												
650942	12	10	2	1	2	2	2	3			2											
651527	7	7	0	1	2	2	2															
651585	11	9	2	1	2	2	1.5	2.5		R4+R5: you are using current_state instead of the State passed to evalF, which causes the unstabilities.	2											
651637	7	7	0	1	2	2	2															
651789	0	0	0																			
652102	12	10	2	1	2	2	2	3			2											
652131	0	0	0																			

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
652335	16	10	6	1	2	2	2	3		Moving cloth stationary points is a lot of fun!	2			3			1					
652649	16	10	6	1	2	2	2	3		Neat use of lambdas!	2		2		2							
652898	0	0	0																			
652937	10	10	0	1	2	2	2	3														
653127	14	10	4	1	2	2	2	3		Your implicit integrator is really just an explicit integrator since the Jacobian is the zero matrix and the solution of Ax=b is trivial.	2								2			
653596	0	0	0																			
653693	10	10	0	1	2	2	2	3		NB: your submission zip was flagged by Windows Defender												
653871	0	0	0																			
653907	10	10	0	1	2	2	2	3														
653910	10	10	0	1	2	2	2	3														
654595	24	10	14	1	2	2	2	3		Particle rendering: textured cloth with shading.	2	3	2	3	2			2				
655057	10	10	0	1	2	2	2	3		R5: non-square shape not significant enough to warrant extra points I'm afraid												
655086	8.5	7	1.5	1	2	2	2			R4: you are not clearing the springs vector at reset() (-0p). Flat shaded cloth mesh.								1.5				
655109	12	10	2	1	2	2	2	3			2											
655251	0	0	0																			
655264	0	0	0																			
655471	10	10	0	1	2	2	2	3														
655691	0	0	0																			
655853	13	10	3	1	2	2	2	3			2							1				
656250	6.5	6.5	0	1	2	2	1.5	0		R4: you are using the class member current_state instead of the supplied argument state. This causes the instabilities.												
656454	16	10	6	1	2	2	2	3			2		2		2							
656616	10	10	0	1	2	2	2	3														
657291	12	10	2	1	2	2	2	3			2											
657314	0	0	0																			
657327	14	10	4	1	2	2	2	3		RK4: not quite right, you are not taking the weighted average of the values k at the end. Spray system: you are not supposed to use the current_state variable in evalF, but rather the function arg state in order to construct a new state that you are free to modify as you wish. You could store the indices of the active particles e.g. in the SpraySystem instance and read from there.	1.5	0.5	2									
657482	0	0	0																			
657796	10	10	0	1	2	2	2	3														
657893	10	10	0	1	2	2	2	3														
659914	12	10	2	1	2	2	2	3			2											
660246	17	10	7	1	2	2	2	3		The collision-extra is quite unstable. Your implicit integrator is actually just an overly complicated and slow explicit integrator. You are never computing the Jacobian (which you should do analytically, not with finite differences!) and essentially just conveniently setting things up such that the result of the Ax=b equation is ~ step*F.	2		2		1				2			

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
660877	0	0	0																			
660893	0	0	0																			
663191	0	0	0																			
663272	0	0	0																			
665380	10	10	0	1	2	2	2	3														
665898	10	10	0	1	2	2	2	3														
666172	12	10	2	1	2	2	2	3			2											
666350	11	9	2	1	2	2	1.5	2.5		R4+R5: you are using the variable current_state in evalF instead of the function argument state, which leads into the instability.	2											
666680	10	10	0	1	2	2	2	3														
667249	10	10	0	1	2	2	2	3														
67137M	0	0	0																			
67627H	15	10	5	1	2	2	2	3			2	3										
677734	0	0	0																			
678089	0	0	0																			
68933B	0	0	0																			
69247N	0	0	0																			
700436	0	0	0																			
705570	69	10	59	1	2	2	2	3			2	3	2	3	2	3	3	4	10	9	18	Adaptive RKF45 (2p), Crank–Nicolson (4p), sparsity patterns (4p), Constraints (8p), GPU sprinkler and GPU integrators (+5 to GPU points)
706566	0	0	0																			
708784	20	10	10	1	2	2	2	3		GPU cloth: evalF writes out of bounds - last if needs to check x < w and y < h (-0p)	2	3								5		
708904	0	0	0																			
708920	12	10	2	1	2	2	2	3							2							
708933	0	0	0																			
709291	5.5	5.5	0	1	2	2	0.5			R4: indexing is off, but general idea seems ok												
709628	12	10	2	1	2	2	2	3			2											
710086	10	10	0	1	2	2	2	3		fSpring: can be replaced with a single expression: `-k * ((p2 - p1).length() - rest_len) * (p2 - p1).normalized()` (handout section 1.2, third equation)												
710497	15.5	10	5.5	1	2	2	2	3		Spary: no collisions	2	2						1.5				
710743	13	10	3	1	2	2	2	3		Wind: looks very artificial, almost no variation over time	2		1									
710976	5	5	0	1	2	2																
711182	5	5	0	1	2	2				R3: bug in first for loop: x0[0] instead of x0[i]												
711467	18	10	8	1	2	2	2	3		Wind: random noise cast to int, which essentially removes all time-dependent variation	2	3	1		2							
711551	12	10	2	1	2	2	2	3			2											
711810	9	7	2	1	2	2	2				2											
711904	0	0	0																			
712550	10	10	0	1	2	2	2	3														
712686	0	0	0																			
712819	6	6	0	1	2	2	1	0		R4: indexing in first for-range loop wrong. R5: positions only not quite enough for points.												
712958	0	0	0																			
713672	0	0	0																			
714985	13	9	4	1	2	2	1.5	2.5		R4, R5: not dividing force by mass to get acc. Generally: please try to match the reference visually (makes debugging and grading easier)	2		2									

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
716080	3	3	0	1		2																
716462	10.5	10	0.5	1	2	2	2	3		RK4: you need to call evalF for the different states, currently they are incorrectly treated as derivatives	0.5											
716718	0	0	0																			
716860	9.5	9.5	0	1	2	2	2	2.5		R5: logic seems fine, probably something with gravity or spring tension												
717377	0	0	0																			
717474	0	0	0																			
717513	12	10	2	1	2	2	2	3			2											
717539	0	0	0																			
718020	0	0	0																			
718871	0	0	0																			
722427	0	0	0																			
723691	6	5	1	1	2	2				RK4: one step too much, final dir given by sum of f0...f3 (weights also don't sum to 6)	1											
723905	0	0	0																			
728667	17.5	10	7.5	1	2	2	2	3			2									5.5		
728900	12	10	2	1	2	2	2	3			2											
729132	0	0	0																			
729967	1	1	0	1	0					R2: no spring force, evalF reads from wrong state indices												
730309	0	0	0																			
732080	0	0	0																			
732255	0	0	0																			
732323	0	0	0																			
732336	0	0	0																			
732352	0	0	0																			
732459	14	10	4	1	2	2	2	3		Sprinkler: no collisions (even though readme claims so?)	2	2										
76509T	12	10	2	1	2	2	2	3			2											
765510	15.5	10	5.5	1	2	2	2	3		Wind: doesn't vary smoothly over time, looks artificial. Pretty unique tearing setup!	2		1.5				2					
766331	0	0	0																			
767042	0	0	0																			
767136	3.5	3.5	0	0.5	1.5	1	0.5			R2: fSpring: using sqr, not sqrt. R1, R3: not looping over points, just evolving first one. R4: broken, but overall idea seems OK												
768504	0	0	0																			
769396	12	10	2	1	2	2	2	3			2											
77388B	10.5	10	0.5	1	2	2	2	3		RK4: need to evaluate forces (evalF) at the intermediate steps rather than just sum the the contributions.	0.5											
779124	0	0	0																			
780058	0.5	0.5	0	0.5						R1: taking two steps instead of one, you need just s0[i] + step*f0[i]												
780346	10	10	0	1	2	2	2	3														
782917	0	0	0																			
783563	0	0	0																			
783709	0	0	0																			
786667	0	0	0																			
78708M	0	0	0																			
787543	0	0	0																			
787640	0	0	0																			
788380	0	0	0																			
788678	0	0	0																			
791982	0	0	0																			
795700	0	0	0																			
795755	0	0	0																			

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
796039	0	0	0																			
804183	12	10	2	1	2	2	2	3			2											
829155	0	0	0																			
838191	0	0	0																			
83873J	0	0	0																			
84308F	0.5	0.5	0	0.5	0	0	0	0		R1: you are returning a state that only works for single particle systems, need a loop here! R2: spring force is ignoring the z component of the 3d vector. Drag force is a function of the velocity of a particle and has nothing to do with springs! You are not computing the location change (passing velocity to the returned state) so the spring system never moves!												
84858E	0	0	0																			
848754	18	10	9	1	2	2	2	3	-1	Wind and sphere collision both make cloth unstable. Mouse drag: somewhat unstable. Tearing: not calibrated, cloth breaks on its own. Particle rendering: particles look no different than by default. NOTE: requirements impossible to grade without manually disabling extras in code (-1p)	2	3	1	1	1		1	0				
875170	12	10	2	1	2	2	2	3			2											
875251	12	10	2	1	2	2	2	3		R5: looks fine to me, just shifted to the side	2											
875303	0	0	0																			
875617	0	0	0																			
876399	0	0	0																			
877107	0	0	0																			
877152	15.5	9	6.5	1	2	2	1.5	2.5		R4, R5: not dividing forces by mass	2	3						1.5				
878591	13.5	10	3.5	1	2	2	2	3		RK4: final direction incorrectly computed (needs weights 1,2,2,1). Tearing: you could e.g. override set_state() and remove springs there	1.5		2				0					
878627	12	10	2	1	2	2	2	3			2											
878889	7.5	6.5	1	1	1.5	2	2			R2: all forces need to be divided by mass. RK4: too many steps (k4, k4_f shouldn't be there)	1											
879105	11	9	2	1	2	2	1.5	2.5		R4, R5: no gravity on last point, not dividing by mass	2											
882134	0	0	0																			
885128	12	10	2	1	2	2	2	3			2											
886648	7	7	0	1	2	2	2			RK4: wrong number of evalFs, evaluated at wrong positions, no linear combination at the end	0											
889645	0	0	0																			
892292	0	0	0																			
898351	8	7	1	1	2	2	2			RK4: step size applied too many times, also slowly diverges for circle example	1											
899130	10.5	9.5	1	1	2	2	2	2.5		R5: not dividing by mass. RK4: step size applied in too many places, also diverges slowly on circle example	1											
900016	15	10	5	1	2	2	2	3			2	3										
901170	0	0	0																			
901196	0	0	0																			
913249	0	0	0																			
913333	0	0	0																			
913346	0	0	0																			

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
913566	11	10	2	1	2	2	2	3	-1	R2, R4, R5: using 'current_state_' instead of 'state', which causes instability (all integrators essentially become euler) (-1p)	2											
915221	0	0	0																			
915250	0	0	0																			
917863	12	10	2	1	2	2	2	3			2											
918150	0	0	0																			
918228	8.5	6.5	2	1	2	2	1.5			R4: using f, not state, when computing drag. RK4: typo: (1/0.6) instead of (1/6.0)	2											
918257	0	0	0																			
918309	0	0	0																			
918396	12	10	2	1	2	2	2	3		Cool-looking wind!			2									
918464	24	10	14	1	2	2	2	3		Tearing and mouse drag pretty well tuned and fun to play with!	2	3	2	3	2		2					
918671	12	10	2	1	2	2	2	3		Code looks pretty clean :)	2											
918875	12	10	2	1	2	2	2	3		R4, R5: drag is typically applied per point based on velocity, not velocity difference. RK4: works fine!	2											
930484	6	6	0	1	2	2	1			R4: apply spring forces to both endpoints instead of adding the difference to the first one. Also better to put drag in the first loop												
932440	12	10	2	1	2	2	2	3			2											
935625	7	7	0	1	2	2	2															
939375	0	0	0																			
942618	14.5	10	4.5	1	2	2	2	3		RK4: weights not quite right, falls towards origin in max speed circle example. Collision: cool visualization and interaction! (+1p)	1.5				3							
k28342	0	0	0																			