Name		ID		Class ID	
Date				Time	

FALLING OBJECT TUTORIAL QUESTIONS

Table of Contents

Name		ID		Class ID	
Date				Time	

1 (Falling Object) Single Decimal Point Data Hand Calculation

Before starting to program it is generally a good idea to first calculate a few cases by hand.

Floor [-]	height [m]	x [m]	time [s]	v [m/s]	time [s]	a [m/ss]	time [s]
Roof	42.5	00.0	00.0	03.8	00.4	10.3	00.7
14	39.5	03.0	00.8	10.0	01.0		
13	36.5	06.0	01.1				
12	33.5	09.0	01.4				
11	30.5	12.0	01.6			✗	✗
10	27.5	15.0	01.8	✗	✗	✗	✗
7	18.5	24.0	02.2	✗	✗	✗	✗
4	09.5	33.0	02.6	✗	✗	✗	✗
G	00.0	42.5	03.0	✗	✗	✗	✗

$$v(t)=\frac{dx}{dt}\approx\frac{\Delta x}{\Delta t}=\frac{(x_2-x_1)}{(t_2-t_1)} \qquad t=\frac{t_2+t_1}{2}$$

Name		ID		Class ID	
Date			Time		

2 (Falling Object) Two Decimal Point Data Hand Calculations

Fill in the table by hand.

		displacement		velocity		acceleration	
Floor [-]	**height [m]**	**x [m]**	**time [s]**	**v [m/s]**	**time [s]**	**a [m/s^2]**	**time [s]**
Roof	42.5	00.0	00.00	**03.85**	**00.39**	**09.45**	**00.67**
14	39.5	03.0	00.78	**09.09**	**00.95**		
13	36.5	06.0	01.11				
12	33.5	09.0	01.36				
11	30.5	12.0	01.57			▨	▨
10	27.5	15.0	01.75	▨	▨	▨	▨
7	18.5	24.0	02.22	▨	▨		
4	09.5	33.0	02.60	▨	▨		
G	00.0	42.5	02.95	▨	▨	**TA Sign**	

Based on the answers you have obtained from your hand calculations, what can you conclude?		**TA Sign**
When you have filled in the table and answered the question, have the TA sign		

Name		ID		Class ID	
Date		Time			

3 (Falling Object) Preparing to Code: UML Diagrams

Based on your hand calculations, summarize the procedure using a UML Diagram.

When you have completed the UML diagram. Ask a TA to stamp.	**TA Stamp**

Name		ID		Class ID	
Date			Time		

4 (Falling Object) ANSI-C Code for a Fixed Point Calculation

4.1 Copy the ANSI-C Code from the View Graphs.

4.2 Fill in the table below using numbers generated by your program.

		displacement		velocity		acceleration	
Floor [-]	**height [m]**	**x [m]**	**time [s]**	**v [m/s]**	**time [s]**	**a [m/s^2]**	**time [s]**
Roof	42.5	00.0	00.00	**03.85**	**00.39**	**09.45**	**00.67**
14	39.5	03.0	00.78	**09.09**	**00.95**	**10.03**	**01.09**
13	36.5	06.0	01.11				
12	33.5	09.0	01.36				
11	30.5	12.0	01.57				
10	27.5	15.0	01.75				
7	18.5	24.0	02.22				
4	09.5	33.0	02.60				
G	00.0	42.5	02.95	▨	▨	**TA Sign**	

Write the C-CODE for solving the problem. After your code compiles, linked and runs without error, fill in the table. Show the code and results to the TA.	**TA Sign**
How can you make the calculation very accurate using fixed-point storage?	

Name		ID		Class ID	
Date				Time	

5 (Falling Object) ANSI-C Code for a Floating Point Calculation

5.1 Convert the Fixed Point Code to Floating Point Code,

i.e. use getFloat() from joninlib.h, change variables and fprintf() accordingly.

5.2 Fill in the table below using numbers generated by your program.

		displacement		velocity		acceleration	
Floor [-]	**height [m]**	**x [m]**	**time [s]**	**v [m/s]**	**time [s]**	**a [m/s^2]**	**time [s]**
Roof	42.5	00.0	00.00	**03.85**	**00.39**	**09.45**	**00.67**
14	39.5	03.0	00.78	**09.09**	**00.95**	**10.03**	**01.09**
13	36.5	06.0	01.11				
12	33.5	09.0	01.36				
11	30.5	12.0	01.57				
10	27.5	15.0	01.75				
7	18.5	24.0	02.22				
4	09.5	33.0	02.60			**TA Sign**	
G	00.0	42.5	02.95	▨	▨		

Write the C-CODE for solving the problem. After your code compiles, linked and runs without error, fill in the table. Finally ask the TA to stamp.	**TA Sign**

Name		ID		Class ID	
Date				Time	

6 (Falling Object) Using Pointers to get Input Data

6.1 Write the C-CODE to implement the input functions. (don't use joninlib.h)

Replace getFloat() with your own code. Your code should prompt the user for two numbers: the distance traveled and time or velocity and time. Thus you will need to use pointers in the subroutine as you want to return 2 pieced of data.

6.2 Use your program to check 1 or two previous calculations

This is to check that after changing your code the calculation remains correct.

When your code compiles, linked and runs without error, and you have filled in the table to check the answers haven't changed, ask the TA to stamp.	**TA Sign**

Name		ID		Class ID	
Date			Time		

7 (Falling Object) Avoiding Repetition, Simplifying Input

7.1 Add repetition (in function main() of mg4ptr.c) to calculate derivatives

Make all modifications in the function main of mg4ptr.c. Call your new file mg5loop.c Note that by using repetition, one does not need to re-enter each piece of data twice.

7.2 Create a text file (mgdt.csv) containing the displacement-time data.

Formatted as specified in your fscanf() statement, e.g.,

```
0.0, 0.0
3.0, 0.78
```

7.3 Run your program using the displacement data you have entered.

Check that after changing your code the calculation remains correct. Since you have placed your data in a file, you can have the C-program read the data from the file. This can be done using the input redirect command. For example,

```
gravity < mgdt.csv
```

7.4 Use your program to help you fill in the table, saving output to a file

Just as one can use input redirection to read data from a file, you can also use output redirection to write data to a file. For example,

```
gravity < mgdt.csv > mgvt.csv
```

will read the displacement data from gravdata.csv and save the calculated velocities to v.csv. One can view the contents of v.csv either by opening as a spreadsheet (i.e. using Microsoft Excel) or one can use the command:

```
type mgvt.csv
```

Since the calculation for acceleration is the same (see UML) as that for velocity, except that the input is velocity rather than displacement, the `mgvt.csv` can be used as input, e.g.,

```
gravity < mgvt.csv > mgat.csv
```

		displacement		velocity		acceleration	
Floor [-]	**height [m]**	**x [m]**	**time [s]**	**v [m/s]**	**time [s]**	**a [m/s^2]**	**time [s]**
Roof	42.5	00.0	00.00	**03.85**	**00.39**	**09.45**	**00.67**
14	39.5	03.0	00.78	**09.09**	**00.95**	**10.03**	**01.09**
13	36.5	06.0	01.11				
12	33.5	09.0	01.36				
11	30.5	12.0	01.57				
10	27.5	15.0	01.75				
7	18.5	24.0	02.22				
4	09.5	33.0	02.60			**TA Sign**	
G	00.0	42.5	02.95	▨	▨		

Name		ID		Class ID	
Date				Time	

8 (Falling Object) Arrays: Separating Input, Calculation, Output

8.1 Modify the existing UML diagram to separate input, calculation and output phases of the program.

	TA Stamp

8.2 Modify the program mg5loop.c to store all variables in arrays

You will need to create arrays to hold displacement-time, velocity-time and acceleration-time (a total of 6 arrays). The steps are as follows

1. Declare the six arrays, i.e., d,t and v,t and a,t – one for each column of table.
2. Use a while (or for) loop to read displacement-time data into 2 arrays, Read from the file mgdt.csv you created last week.
3. Use a for loop to calculate average velocities and associated times
4. Use a 2nd for loop to calculate average accelerations and associated times
5. Use a 3rd for loop to print out a table similar to that below.:

x [m]	time [s]	v [m/s]	time [s]	a [m/s/s]	time [s]
00.0	00.00	03.85	00.39	09.45	00.67
03.0	00.78	09.09	00.95	10.03	01.09
...	...	...	...	...	...

Note that in this work you should only modify the function main(). Do not modify any other function in the program. (without arrays, this would not be possible).

8.3 Save your velocity and acceleration calculations to a file

Last week we saw that since you have placed your data in a file, you can have the C-program read the data from the file and write to another file by combining input and output redirection. Recall that to do both at once we can write:

```
mg6arr < mgdt.csv  > mgtable.csv
```

8.4 Open output file "table.csv" in Microsoft Excel or OpenOffice Calc

After you have opened your output file in the spreadsheet program, ask the TA to stamp. He will check to see that your table is formatted correctly and contains the correct values.	**TA Sign**

Name		ID		Class ID	
Date				Time	

9 (Falling Object) Records: Linking both coordinates

9.1 Modify the program mg6arr.c to link data. Call new program mg7rec.c.

9.1.1 Define a new type, Data, that links each quantity with its time.

9.1.2 Declare three arrays of records of type data

1. Replace the displacement and time arrays with single array of records
2. Replace the velocity and velocity time arrays with single array of records
3. Replace the acceleration and acceleration time arrays with single array of records

9.1.3 Modify the rest of function main() accordingly.

9.1.4 Check to make sure that your program is running correctly

Use the same input file as you used last week in testing your array program. In other words get the data from the same file but save it to another file, e.g.

```
mg7rec < mgdt.csv  > tablerec.csv
```

Compare the output file this week with the file generated by the array program. They should be the same.

When your code compiles, linked and runs without error and you have verified your calculations, ask the TA to stamp. If this is being done as a homework assignment, please copy and print your code on the back of this sheet of paper.	**TA Sign**

9.2 Create a new function getRec() to replace getFloat().

9.2.1 Create a new function prototype

Your new function should not use pointers but use “return” to get data back to your main function. For getFloat() we used two pointers.

```
void getFloat(double *x, double *t);
```

The function prototype of the new function should be.

```
struct Data getRec(void);
```

9.2.2 Write the function definition for your new function getRec()

9.2.3 Replace the function call to getFloat() with one to getRec()

9.2.4 Check to make sure that your program is running correctly

Use the same input file as you used last week in testing your array program. In other words get the data from the same file but save it to another file, e.g.

```
mg7rec < mgdt.csv  > tablerec.csv
```

Compare the output file this week with the file generated by the array program. They should be the same.

When your code compiles, linked and runs without error and you have verified your calculations, ask the TA to stamp. If this is being done as a homework assignment, please copy and print your code on the back of this sheet of paper.	**TA Sign**

Name		ID		Class ID	
Date			Time		

10 Appendix

Velocity Equation

$$v(t)=\frac{dx}{dt}\approx\frac{\Delta x}{\Delta t}=\frac{(x_2-x_1)}{(t_2-t_1)}$$

$$t=\frac{t_2+t_1}{2}$$

Acceleration Equation

$$a(t)=\frac{dv}{dt}\approx\frac{\Delta v}{\Delta t}=\frac{(v_2-v_1)}{(t_2-t_1)}$$

$$t=\frac{t_2+t_1}{2}$$

Calculation Example – Velocity 1

$$v\left(\frac{0.8+0.0}{2}\right)=\frac{3-0}{0.8-0} \rightarrow v(0.4)=3.75=3.8$$

Calculation Example Velocity 2

$$v\left(\frac{1.1+0.8}{2}\right)=\frac{6-3}{1.1-0.8} \rightarrow v(1.0)=10.0$$

Calculation Example Acceleration

$$a\left(\frac{1.0+0.4}{2}\right) = \frac{10-3.8}{1.0-0.4} \rightarrow a(0.7)=10.3$$