

Program 2, CS 101, Prof. Loftin

Due Monday, February 23, 2004

The problem

A moving car is being braked at a constant acceleration. Compute whether or not the car has come to a complete halt. Read v_0 (the initial velocity), a (the acceleration—which is negative to represent braking), and t (the amount of braking time) from the keyboard. Use appropriate prompts to for each of these. You should note in your prompts that v_0 and t should be positive, and a should be negative. If the quantity $t|a|$ (t times the absolute value of a) is greater than or equal to the initial velocity v_0 , the car has come to a complete stop; otherwise the final velocity is $v_0 + at$. Print the final velocity or the message, “The car has come to a complete stop.”

Next, your program should print out the distance the car has travelled while braking. If the car has come to a complete stop, this distance is given by

$$\frac{v_0^2}{2|a|}.$$

On the other hand, if the car has not come to a complete stop, the distance the car has travelled is given by

$$v_0t + \frac{at^2}{2}.$$

Sample runs

Here are two sample runs of my program:

```
<pegasus> a.out
Enter an initial velocity (a positive number): 10
Enter an acceleration for braking (a negative number): -2
Enter the time allowed for braking (a positive number): 3
```

```
The final velocity is 4.
The distance the car has travelled while braking is 21.
<pegasus>
```

```

<pegasus> a.out
Enter an initial velocity (a positive number): 8
Enter an acceleration for braking (a negative number): -5
Enter the time allowed for braking (a positive number): 2

The car has come to a complete stop.
The distance the car has travelled while braking is 6.4.
<pegasus>

```

How to do it

You should use an `if else` statement to separate the two cases.

Hints

The square of a number `x` may be represented by `x * x` or by `pow(x,2)`. (The function `pow` is contained in the header file `math.h` or `cmath`.)

The variables `v0`, `a` and `t` should be `double` type. Recall that the absolute value of a `double` variable `v0` is given by `fabs(v0)`. You will need to include one of the header files `math.h` or `cmath` to use the function `fabs`. You probably also want to use variables `v_final` and `dist` to represent the final velocity and the distance travelled while braking.

Handing in your program

Send it your program to me as an attachment to an email message. Send it to `loftin@pegasus.rutgers.edu`.

- DO NOT SEND IT TO MY `andromeda` ACCOUNT.
- The file you attach should be something like `Program2.cpp` (assuming that's what you've named your program). DO NOT SEND A FILE WITH A `~` AT THE END OF IT (such as `Program2.cpp~`). This is an old version of your program that `emacs` has saved for you. It will likely include errors that you fixed the last time you edited the file.
- On the `Cc:` line in pine, put your own email address. This will send you a copy of what you send to me. (So this way if there is some problem with the attachment you've sent—if you've sent the wrong file or your program isn't attached for some reason—you can tell and send me the correct version.)
- Turn in your program by any time on Monday.

Program requirements

Your program must compile on `pegasus` using our usual `g++` compiler.

Programs which do not compile will not be graded. Your program should be documented with meaningful variable names, and comments where necessary. Your program should begin with a comment giving your name and a brief description of the program. Your program must contain:

- Meaningful prompts for the user to enter the initial velocity, acceleration and time. These should include instructions that the initial velocity and time should be positive, and the acceleration should be negative.
- You may assume the user will follow these instructions (so you do not have to print an error message if for example the user enters a positive number for the acceleration).

Running my solution

You can run my solution by typing

```
~loftin/cs101/solution2
```

at the `<pegasus>` prompt.