Programming Assignment 2
It's Not Fare!

Time due: 11:00 PM Tuesday, October 17

Before you ask a question about this specification, see if it has already been addressed by the Project 2 FAQ. And read the FAQ before you turn in this project, to be sure you didn't misinterpret anything.

(Be sure you also do the homework accompanying this project.)

You are a software developer for the Lost Angels County Department of Transportation. Bus riders in the county may purchase fare tickets. You have been assigned to write a program that prints these tickets.

Your program must accept as input the rider's destination, the number of zone boundaries crossed, the rider's age, and whether or not the rider is a student. The output will repeat the destination and tell the fare.

Here is an example of a dialog with the program (user input is shown in **boldface**):

```
Age of rider: 21
Student? (y/n): n
Destination: Bay City
Number of zone boundaries crossed: 2
---
The fare to Bay City is $2.45
```

The fare is $1.35 plus $0.55 per zone boundary crossed, unless a discount applies:

- Riders under 18 get a break for short trips: For trips crossing 0 or 1 zone boundaries, the fare is $0.65
- For trips crossing 0 or 1 zone boundaries, students 18 or over get the same break as riders under 18.
- The senior fare for riders 65 and over is $0.55 plus $0.25 per zone boundary crossed. However, for a trip crossing 0 boundaries, the senior fare is only $0.45. Seniors who believe in lifelong learning (i.e., are students) are no dummies: They pay the cheapest fare available to them.

Here's another example:

```
Age of rider: 22
Student? (y/n): y
Destination: Jollywood
Number of zone boundaries crossed: 1
---
The fare to Jollywood is $0.65
```

You can test your understanding of the fare schedule by experimenting with the fare calculator we found at the Lost Angels County Department of Transportation website.

Your program must collect the information for one transaction in the manner indicated by the examples, and then write to `cout` a line with three hyphens only (no spaces or other characters), followed by exactly one line in a format required below. Our grading tool will judge the correctness of your program by examining only the line following the line with three hyphens (and verifying that there are no additional lines). That one line you write must be in one of the following five forms; the text must be **identical** to what is shown, except that *italicized* items are replaced as appropriate:

- If the rider's age is negative:
  `The age must not be negative`
- If the student status is not `y` or `n` (must be lower case):
  `You must enter y or n`
- If nothing was provided for the destination:
  `You must enter a destination`
- If the number of zone boundaries crossed is negative:
  `The number of zone boundaries crossed must not be negative`
- If the input is valid and none of the preceding situations holds:
  `The fare to` *`destination`* `is` *`$amount`*

In the last case, *destination* must be the destination as entered, and *amount* must be the correct answer, shown as a non-negative number with at least one digit to the left and exactly two digits to the right of the decimal point. (See p. 32-33 of the Savitch book.) The lines you write must not start with any spaces. If you are not a good speller or typist, or if English is not your first language, be especially careful about duplicating the messages **exactly**. Here are some foolish mistakes that may cause you to get no points for correctness on this project, no matter how much time you put into it:

- Not writing to `cout` a line with exactly three hyphens in *all* cases.
- Writing any spaces on the line that is supposed to have three hyphens.
- Writing more than one line after the line with three hyphens. Don't, for example, add a gratuitous "Thank you for riding a Lost Angels County bus."
- Writing lines to `cerr` instead of `cout`.
- Writing lines like these:
- `The fair to Redundant Beach is $1.05` *misspelling*
- `The Fare to Redundant Beach is $1.05` *wrong capitalization*
- `The fare for Redundant Beach is $1.05` *wrong text*
- `The fare to Redundant Beach is $ 1.05` *extra space*
- `The fare to Redundant Beach is $1.05.` *extra period*
- `The fare to Redundant Beach is $1.050` *extra digit*
-

Your program must gather the age, the student status, the destination type, and the number of zone boundaries crossed in that order. However, if you detect an error in an item, you do not have to request or get the remaining items if you don't want to; just be sure you write to `cout` the line of three hyphens, the required message and nothing more after that. If instead you choose to gather all input first before checking for any errors, and you detect more than one error, then after writing the line of three hyphens, write only the error message for the earliest erroneous item.

You will not write any loops in this program. This means that each time you run the program, it handles only one bus ticket transaction. It also means that in the case of bad input, you must not keep prompting the user until you get something acceptable; our grading tool will not recognize that you're doing that.

A string with no characters in it is the empty string. A string with at least one character in it is not the empty string, even if the only characters in it are things like spaces or tabs. Although realistically it would be silly to have a destination name consisting of seventeen spaces and nothing more, treat that as you would any other non-empty string: as a valid name. (Since you don't yet know how to check for that kind of situation anyway, we're not requiring you to.)

The one kind of input error your program does **not** have to deal with is the case of not finding an integer in the input where an integer is expected. Our grading tool will not, for example, supply the text `Granola Hills` or the number `23.456` when your program requests a rider's age.

The correctness of your program must not depend on undefined program behavior. Your program could not, for example, assume anything about `n`'s value at the point indicated:

```
int main()
{
    int n;
    int m = 42 * n;  // n's value is undefined
    …
```

What you will turn in for this assignment is a zip file containing these three files and nothing more:

1. A text file named **fare.cpp** that contains the source code for your C++ program. Your source code should have helpful comments that tell the purpose of the major program segments and explain any tricky code.
2. A file named **report.docx** or **report.doc** (in Microsoft Word format) or **report.txt** (an ordinary text file) that contains:
    a. A brief description of notable obstacles you overcame. (In Project 1, for example, some people had the problem of figuring out how to work with more than one version of a program in Visual C++.)
    b. A list of the test data that could be used to thoroughly test your program, along with the reason for each test. You don't have to include the results of the tests, but you must note which test cases your program does not handle correctly. (This could happen if you didn't have time to write a complete solution, or if you ran out of time while still debugging a supposedly complete solution.) For Project 1, for example, such a list, had it been required, might have started off like this:

    More surveyed than the total expressing approval and disapproval (1000, 413, 382)
    Fewer surveyed than the total expressing approval and disapproval (500, 413, 382)
    Zero people surveyed (0, 100, 100)
    Data giving a non-integer percentage (1000, 413, 382)
    More expressing approval than disapproval (1000, 413, 382)
    Equal numbers expressing approval and disapproval (1000, 500, 500)
    …

3. A file named **hw.docx** or **hw.doc** (in Microsoft Word format) or **hw.txt** (an ordinary text file) that contains your solution to the homework accompanying Project 2.

By October 16, there will be links on the class webpage that will enable you to turn in your zip file electronically. Turn in the file by the due time above. Give yourself enough time to be sure you can turn something in, because we will not accept excuses like "My network connection at home was down, and I didn't have a way to copy my files and bring them to a SEASnet machine." There's a lot to be said for turning in a preliminary version of your program, report, and homework early (You can always overwrite it with a later submission). That way you have something submitted in case there's a problem later. Notice that most of the test data portion of your report can be written from the requirements in this specification, before you even start designing your program.

The writeup Some Things about Strings tells you what you need to know about strings for this project.

As you develop your program, periodically try it out under another compiler (g31 on cs31.seas.ucla.edu if you're doing your primary development using Visual C++ or Xcode). Sometimes one compiler will warn you about something that another is silent about, so you may be able to find and fix more errors sooner. If running your program under both environments with the same input gives you different results, your program is probably relying on undefined behavior (such as using the value of an uninitialized variable), which we prohibit.