Tax Calculator
In this project, you will be writing a program to compute the taxes for an individual based on their income, deductions, and filing status. This project focuses on files, vectors, and maps.

Program Features
You will be writing a program in C++ to compute the taxes for an individual based on their income, deductions, and filing status. We want our program to be general and reusable, but tax rates often change from year to year, so to accommodate this **we will read all of the tax bracket information from a file.** With this approach, the program itself doesn't need to be changed when the tax rates change.

Examples Using 2009 Tax Brackets
The table below contains the tax brackets for 2009, using these rates we will demonstrate the process of calculating an individual's tax bill. As stated above your program will read the tax bracket information from a file, **so you cannot hard code these values into your program. They will change when we test your program!**

2009 U.S. Federal Personal Tax Rates

Marginal Tax Rate	Single (S)	Married Filing Jointly (M)	Married Filing Separately (F)	Head of Household (H)
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351– $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 - $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

There are 4 filing statuses: Single (S), Married (M), married Filing separately (F), and Head of household (H). For someone with **Single** filing status and a **taxable income** of $10,000, the first $8,350 is taxed at 10% and the other $1,650 is taxed at 15%. The total tax bill would be $1,082.50:

= ($8,350.00 * 0.10) + (($10,000.00 - **$8,350.00**) * 0.15)
= $835.00 + ($1,650.00*0.15)
= $835.00 + $247.50
= **$1,082.50**

For someone with **Head of Household** filing status and a **taxable income** of $150,000, the total tax bill would be $33,329.00.

= ($11,950 * 0.10) + (($45,500 - **$11,950**) * 0.15) + (($117,450 - **$45,500**) * 0.25) +
(($150,000 – **$117,450**) * 0.28)
= 1195 + 5032.50 + 17987.50 + 9114
= **$33,329.00**

Sample Program Execution
The user should be prompted for a file containing the tax bracket information and then a single uppercase letter indicating the filing status (S, M, F, or H). The user should then be prompted to enter a **gross income** followed by **deductions** on a single line. The **taxable income** (used above to compute the tax bill) is **gross income – deductions**.

```
Welcome to the CS1064 Tax Calculator program!
Enter the file storing the tax brackets: taxinfo2009.txt
Enter filing status (S/M/F/H): M
Enter Total Income and Deductions: 19700 3000
Total Tax: $1670.00
Would you like to calculate another person's taxes? (yes/no): yes
Enter filing status (S/M/F/H): S
Enter Taxable Income and Deductions: 43444 4300
Total Tax: $5973.50
Would you like to calculate another person's taxes? (yes/no): no
```

While this example only shows the Single (S) and Married (M) tax filing statuses, your program should be able to compute the correct tax bill for any of the 4 statuses (S, M, F, or H) listed above. After finishing each tax calculation, the user should be prompted as to whether they'd like to calculate another person's taxes. The program should continue until the user says "no" and should be able to compute an arbitrary number of tax bills.

Make sure the prompts are as close as possible to the example above; also notice the formatting for the output. Note that the total tax **includes a dollar sign and is printed to exactly 2 places**.

Tax Brackets File Format
The file in the sample execution above (`taxinfo2009.txt`) contains the tax rates and tax brackets for a particular year, and formatted is like the text shown below:

```
10.0 15.0 25.0 28.0 33.0 35.0

S
8350 33950 82250 171550 372950

M
16700 67900 137050 208850 372950

F
8350 33950 68525 104425 186475

H
11950 45500 117450 190200 372950
```

The first line of numbers contains the tax rate for each bracket. After that, there will always be four filing statues in the file but **their ordering may change.** Each status is made up of a line with a letter (the status), and then a line with the tax bracket values. Further, there is a blank line between each filing status.

Each number listed for a status is the upper bound for a bracket, for example: with filing status `S`, all income `<= 8350` is taxed at 10% (`10.0`). **Note:** there's one upper bound "missing" from the file, there's **no upper bound provided for the highest rate**. Anything above the final upper bound should be taxed at the highest rate, i.e. anything less than infinity is taxed at the top rate.

All input files will follow this format and your program will not be tested with improperly formatted files.

Here's another example with the tax brackets for 2014 (`taxdata2014.txt`):

```
10.0 15.0 25.0 28.0 33.0 35.0 39.6

M
18150 73800 148850 226850 405100 457600

S
9075 36900 89350 186350 405100 406750

H
12950 49400 127550 206600 405100 432200

F
9075 36900 74425 113425 202550 228800
```

While the overall file format is the same, (rates then statuses and brackets) there is an **additional tax rate and bracket:** `39.6`. So there are 7 tax rates and each status has 6 provided brackets. When testing your program there may be additional (or fewer) rates and brackets, but there will always be a consistent number of rates and brackets within a single file, e.g. 6 rates and 5 brackets or 7 rates and 6 brackets.

Program Strategy and Requirements

One technique that is helpful in solving problems (whether they are programming problems or otherwise) is decomposition: breaking the problem into smaller and manageable pieces. Here are some helpful hints for breaking down the program:

First, make sure you understand how a single calculation is performed. Start by doing several calculations by hand. Then use "Stepwise Refinement" and decompose the program, perhaps into the following phases:

- Input
- Calculations
- Output

Write the Calculations code in steps: Figure out how to do the calculation for Single, $8,000 and write the code (and test it), then figure out the calculation for Single, $10,000 (and test it).

Since the number of rates and brackets can change from program run to program run, it isn't possible to hardcode the bracket values into the program, and you **won't be able to solve the problem with a giant if/elif/else statement**. You will still need if/else statements though.

Instead you'll want to store the rates and the brackets in `vector`s, and use them to compute the tax bill. Once you know the status, you want to move through the appropriate `vector` computing the tax as you go. As you move through the `vector` there will be two possibilities: (1) the taxable income is greater than the bracket upper bound (`150000 > 11950.00`) or (2) the bracket bound greater than (or equal to) the taxable income (`190200.00 >= 150000`). You're going to do something slightly different in each case and I'd review the examples at the beginning of this document for hints. Case 2 is your stopping point, when you reach that point you don't need to look at the next bracket.

So you'll need some sort of loop (to move through the `vector`) and probably an if/else statement (for the 2 cases) to compute the taxes.

Hints

For this project, I'd recommend using string streams, `vector`s, and `map`s. For example, the function below will read the brackets from a line in the file and put them in a `vector` containing `double`s.

The code below that shows how you might use a `map` to store both the filing status and the brackets.

```
// Function to read the brackets from a line.
vector<double> get_brackets(string line)
{
     double bound;
     vector<double> brackets;
     istringstream ss(line);

     while (ss >> bound)
     {
          brackets.push_back(bound);
     }
     // Add an additional upper bound for infinity.
     // Makes calculations a little easier...
     brackets.push_back(numeric_limits<double>::infinity());

     return brackets;
}
...
```

```
// In the main() function, we use a map of char to vector<double>:
map<char, vector<double> > status_to_brackets;

// status and line come from the input file
status_to_brackets[status] = get_brackets(line);

...

// Now I can I get the brackets using the map and the user provided
// filing status:

cout << "Enter filing status: ";
cin << s;

// Iterate through the vector with the brackets.
for (int b = 0; b < status_to_brackets[s].size(); b++)
{
     // Do something with status_to_brackets[s][b]
}
```

Documentation and Style
This project **will have a portion of its grade devoted to documentation and code style.** Refer to the style guidelines on the course website for examples of what the TAs will be looking for.

This assignment is worth **100 points**. When you submit, Web-CAT will test that your program works correctly and creates the proper output, if your program works correctly you will receive **80 points** from the automated testing provided by Web-CAT. The remaining **20 points** are for style and documentation, which will be manually graded by the TAs, using the rubric posted on the course website.

What to Submit
For this assignment you should submit your `main.cpp` file from inside of your project directory.

This assignment will be graded automatically. Test your programs thoroughly before submitting them. Make sure that your programs produce correct results for every logically valid test case you can think of. Do not waste submissions on untested code, or on code that does not compile with the supplied code from the course website.

Web-CAT will assign a score based on runtime testing of your submission; your best score will be counted; the TAs will later verify that your best submission meets the stated restrictions, and assess penalties if not.

To submit this assignment:
1. Visit http://web-cat.cs.vt.edu in your web browser.
2. Enter your Virginia Tech PID and password in the appropriate fields on the log-in screen, and make sure that **Virginia Tech** is selected as the institution. Click **Login**.

3. The Web-CAT home screen will display useful announcements and assignments that are currently accepting submissions. Find the assignment that you want to submit in the table, and click the "Submit" button next to it.
4. Click the **Browse...** button and select the file you want to upload. The homework assignments and programming projects for this course should be self-contained in a single **main.cpp** file, so you can simply select that one file.
5. Click the **Upload Submission** button. The next page will ask you to review your selection to ensure that you have chosen the right file. If everything looks correct, click **Confirm**.

The next page will show that your assignment is currently queued for grading, with an estimated wait time. This page will refresh itself automatically, and when grading is complete you will be taken to a page with your results.

Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the submitted file:

```
//    On my honor:
//
//    - I have not discussed the C++ language code in my program with
//      anyone other than my instructor or the teaching assistants
//      assigned to this course.
//
//    - I have not used C++ language code obtained from another student,
//      or any other unauthorized source, either modified or unmodified.
//
//    - If any C++ language code or documentation used in my program
//      was obtained from an allowed source, such as a text book or course
//      notes, that has been clearly noted with a proper citation in
//      the comments of my program.
//
//    <Student Name>
```

Failure to include this pledge in a submission will result in the submission being disallowed during code review.