Due Thursday, 2000 Jan 20, at the beginning of class.

1 Reading

Read chapter 1 of the textbook.

2 Programming Tips

2.1 Software Design Tip

Unlike writing essays, it is frequently easier to build programs by iteratively adding features. If you test after each adding each feature, any mistakes are likely to be in the newly added code.

2.2 Xemacs Tip

To edit a file called "foo.cc" using the text editor Xemacs, type `xemacs foo.cc &` in any shell. Omitting the filename just starts Xemacs. Omitting the ampersand (&) will prevent the shell from being used until Xemacs finishes; this may be the right behavior when using Xemacs via telnet.

2.3 Compilation Tip

Use the g++ `-Wall -pedantic` compiler options to produce warning messages. This may help find programming mistakes more easily than the default of producing just using syntax errors.

Use the g++ `-o output-file-name` compiler options to name the resulting executable "output-file-name."

For example, one could use

```
g++ -Wall -pedantic foo.cc -o foo
```

to compile a C++ file named "foo.cc" and create an executable called "foo," rather than "a.out."

3 Submission Rules

Please submit your homework via email to cs1321-1@cs.trinity.edu or to cs1321-2@cs.trinity.edu, according to your section. Please read the instructions.

For this homework, please email only your source code file, not any test files, e.g., a file of sales tax rates. There is no need for a Makefile.

4 Problems

Write a program to compute the correct state sales tax, given a list of item prices. For example, a retailer selling two items each costing \$3.45 and one item costing \$123.45 could compute the sales tax by typing

```
salesTax salesTaxRates TX 3.45 3.45 123.45
```

into any shell.

`salesTax` is the name of the executable program.

`salesTaxRates` is the name of a file with each line containing a two-letter state abbreviation, e.g., TX and CA, and the state sales tax rate written as a percentage, e.g., 6.25 for Texas and 7.25 for California. Sales tax rates are stored in a file because they continually change.

`TX` indicates the state for which the sales tax should be computed.

`3.45 3.45 123.45` indicates the item prices, written as decimal numbers with two digits after the decimal point.

The program should print 8.15, which is the sales tax due. That is,

$$(3.45 + 3.45 + 123.45) * 6.25\% = 8.146875,$$

which is then rounded to the closest penny.

Details:

- Use command-line arguments. Command-line arguments are yet another way for programs to obtain input from the user. When running many programs we wrote in CS1320, we prompted the user for necessary input such as filenames. Instead, any input known when the program begins can be sent to the program using command-line arguments.

  The example given above has six command-line arguments, separated by whitespace. Just as for arrays, they are numbered starting with zero. For example, `argv[2]` is `TX`, and `argv[5]` is `123.45`.

  Here are the three rules for using command-line arguments:

  1. The `main` function should have two parameters:
     `int main(int argc, char *argv[])`.
  2. To access the ith argument, use `argv[i]`.
  3. `argv[argc]` equals 0.

  The first parameter `argc` indicates the number of command-line arguments. The second parameter can be interpreted as an array (`[]`) of C-style strings, i.e., strings ending with a null-termination character. For example, `argv[0]` is a C-style string listing the executable program's name, e.g., "salesTax." For an example program, see an echo program.

- All command-line arguments are C-style strings, even for numeric arguments, e.g., `argv[3]` is the string "3.45". Tip: Use the strtod() library function to convert from a string to a `double`. Using

  ```
  double d = strtod(argv[3], static_cast<char **>(0))
  ```

  assigns the double value 3.45 to the variable `d`. Be sure to include stdlib.h.

- Round fractions of pennies to the nearest penny. Round half-pennies up. For example, 10.5 cents is rounded to 11 cents, but 10.49999 cents is rounded down to 10 cents.

- Be sure your program works for any number of item prices and for a file of sales tax rates of any length. Be sure your program checks for enough command-line arguments and works if the desired state is not found in the sales tax file.