Northwestern University
Department of Computer Science

Programming Assignment 1

CS 211 Introduction to Programming II
Spring Quarter 2006

Out : Monday 04/03/06, Due : Thursday 04/10/06 at 11:59pm

..... THERE ARE THREE PAGES TO THIS HANDOUT .....

Goals of this assignment

The main purpose of this assignment is to introduce you to the basics of C++ programming and the g++ compiler. After you complete the assignment, you should be able to

- Understand some of the usual error messages of the compiler.
- Declare and define variables
- Write basic expressions and statements
- Perform basic I/O
- Declare and implement your own classes
- Create and uses instances of classes

Provided files

Download the file `pa1.tgz` or `pa1.zip` from the webpage. These are archives of starter code for this assignment. `pa1.tgz` is created using UNIX's `tar` command. `pa1.zip` is a WinZip archive. They decompress to a directory named `1` with two files: `Makefile` and `problem1.cpp`

Problem 1: Compiler warnings/errors

If you look in the file `Makefile` you'll see a rule for p1 that depends only on `problem1.o`. Default rules in `make` will generate `problem1.o` from `problem1.cpp` using `g++`. The make variable `CPPFLAGS` gets sent to the invocation of `g++`. We use this to get some more explicit error messages. As specified, the rule will generate an executable of the same name as the make target, `p1` in this case. However, since the code is erroneous, no executable will be generated at this time. *If you use a development environment other than UNIX, you're on your own to figure out how to compile and build the code.*

The program is supposed to loop repeatedly requesting an integer `n` and calculating `fib(n)`. However, it is riddled with errors (five to eight, depending on how you interpret) that can be detected. By examining the output from building `p1` correct the errors so that the problem runs correctly.

Download the file `problem1.cpp` from the course webpage Compile the broken file. Save the error and warning messages in a text file. An easy way to do that is to redirect the error output, like this:

```
make 2> errors1.txt
```

What to do:
Correct the code so that it compiles successfully with no errors, warnings, or logic errors. Once that is done, you should be able to run your program by typing:

```
./p1
```

Here's the process you should use to complete this part of the assignment:

1. Open `errors1.txt` with a text editor, e.g., Notepad or emacs or vi or whatever, but not Word. Add comments to each error message, describing in your own words,

   - what line of code the error is referring to
   - what the error message says is wrong with that line of code
   - what change you're making to that line

   If you don't understand an error message, or how to fix it, say that.

2. Fix the errors you do understand in `problem1.cpp`. If you see logic errors in the code as you go along, fix those too. Write those changes down in a file called bugs.txt, describing

   - what the mistake is
   - how it affects what the program produces
   - what change you're making

3. Re-compile the fixed code. New error messages will probably appear. This does not necessarily mean that you made a mistake. Many errors are "hidden" by previous mistakes. Save these messages in a new file, `errors2.txt`. Repeat the above process of explaining and changing, until the code compiles.

4. Compile and link the code into an executable and run it. Study the output and fix any remaining logic errors until it produces the correct output. Document these changes in bugs.txt.

5. Combine your syntactic error files into one file `errors.txt`.

What to submit :

- File `problem1.cpp` containing the corrected program.
- A text document (not Word) `errors.txt` containing a list of the errors/warnings and your interpretation of what each one means.
- A text document (not Word) `bugs.txt` containing a short explanation of any logic errors you found and corrected, or a statement that there aren't any.
- Collect all of these files into an archive named `problem1_1.zip` or `problem1_1.tgz`

Problem 2: A Simple C++ Class

Do Exercise 3.14 of Deitel and Deitel, implementing an `Employee` class. Following good programming practice, break your class into three files:

- `Employee.h`, which should have just the declaration for the `Employee` class, not the implementation.
- `Employee.cpp`, with the implementation of the `Employee` class
- `EmployeeTester.cpp`, this should have your `main` function and code to test your `Employee` class

What to submit :

- The files `Employee.h`, and `Employee.cpp`, `EmployeeTester.cpp`
- Collect all of these files into an archive named `problem1_2.zip` or `problem1_2.tgz`

Submission guidelines

Write your name and email in the form of comments at the beginning of every file you submit.

You will need to combine the results of each part of your assignment into a tarball (compressed tar file) or a zip file. Use the digital drop box on Blackboard to submit your homework.

Start your work early and take advantage of the newsgroup, `cs.211` and office hours.