CS113 Assignment 1 — Introduction to C, Fall 2007
Due Wednesday January 30, 2008, 11:59:59PM

0 Introduction

Do <u>two</u> of the following three problems.

This assignment will help you learn the syntax of C. It will also help you learn how to do basic operations with C, like console I/O and mathematical calculations.

You may work with a partner on this assignment, or you may work alone. You must abide by the academic integrity standards of CS 113, the Computer Science department, and Cornell University. Links to these academic integrity policies are provided on the course website. In particular, you may not share code with anyone who is not your partner.

Submit your solutions by the deadline through CMS. The last section of this assignment summarizes the files that you need to submit. If you work with a partner, set up your partnership on CMS. You only need to submit one copy of the code.

Remember to use good programming style. In particular, use meaningful variable and function names. Add comments to clarify your code, and include a comment at the top of the source file that includes your name, NetId, the date, and a brief description of your solution. Correct solutions that are short, simple, and efficient are superior to submissions that are unnecessarily long, complex, or slow. All code that you submit must compile and run without warnings or errors.

This assignment will be graded out of 100 points and is worth 25% of your course grade.

Advice: **Start early!**

1 Retirement planning

It's never too early to start thinking about retirement! In this problem you will develop a simple program to help plan for retirement savings. We will assume the following simple scenario:

1. You begin saving for retirement at age a_1 and stop saving at age a_2.
2. For the years when you are between ages a_1 and a_2, you add d dollars to your retirement account each month.
3. You retire at age a_3, and begin withdrawing w dollars per month from the account.
4. Your retirement account earns an annual interest rate i. Interest is compounded monthly.

Write a program that displays the predicted balance in the retirement account for every fifth year between age a_1 and age 100. The program should first ask the user to enter values for a_1, a_2, a_3, d, w, and i. The program may assume that the user enters valid inputs; that is, your program need not check for input errors. The output of the program should be a well-formatted table.

Here is what a sample execution of the program might look like:

```
Enter age at which saving begins (a1): 30
Enter age at which saving ends (a2): 40
Enter monthly savings amount: 100
Enter age at which retirement begins (a2): 65
Enter interest rate: 0.1
```

```
Enter monthly withdrawal amount: 2500

Age      Account value
---      -------------
 30            1267.03
 35            9892.89
 40           24085.10
 45           39627.43
 50           65199.36
 55          107273.10
 60          176497.38
 65          290392.72
 70          282579.79
 75          269725.10
 80          248575.16
 85          213776.98
 90          156523.23
 95           62323.11
100               0.00
```

Your output format need not match this example exactly. If you need to make assumptions (e.g. whether deposits are made at the beginning or end of the month, etc.), make a note of it in a comment in your code. Submit your solution as `retire.c`.

Side note: Even though this model is simple (we haven't considered the effects of inflation or taxes, for example), the program is still illustrative. Compare what happens if you save \$100/month for just ten years between ages 20 and 30, versus saving \$100/month for *thirty-five* years between ages 30 and 65.

2 Integer Intelligence

You have just been hired by the CIA's secretive Non-negative Integer Reconnaissance Division (NIRD). Your first assignment is to create a program that can instantly display known intelligence about any positive integer.

Write a program that asks the user to type an integer, wait for the input, and then check to make sure it is in the range $[1, 10000]$. You may assume that the user's input is actually an integer; for example, the program does not need to check whether the user entered letters. The program must analyze the integer n and output the following information:

- whether n is even or odd
- how many decimal digits n has
- the sum of n's decimal digits
- a list of n's factors
- whether or not n is prime.
- whether or not n is a *perfect square*. A perfect square is an integer whose square root is also integral.
- its binary (base 2) representation
- its octal (base 8) representation
- its *iterated logarithm*, $\log^* n$. The iterated logarithm is defined as follows:

$$\log^* n = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \log^*(\log n) & \text{otherwise} \end{cases}$$

Here is what a session with the program might look like:

```
*TOP SECRET*   NIRD INTELLIGENCE SYSTEM   *TOP SECRET*

Please enter a non-negative integer: -3
Error: -3 is not a non-negative integer! Please try again.

Please enter a non-negative integer: 12

even or odd         : even
# of decimal digits : 2
sum of digits       : 3
list of factors     : 1 2 3 4 6 12
perfect square      : no
primality           : not prime
binary              : 1100
octal               : 14
log*                : 2
```

(Your output need not match this format exactly.)

Submit your solution in a file called `intel.c`.

3 Typo correction

Modern word processing programs like Microsoft Word include spelling and grammar checkers to help detect and correct typographic errors. These algorithms have become quite complex, including large dictionaries of common errors and sophisticated natural language processing (NLP) analysis techniques.

In this problem, you will develop a program to detect errors using a much simpler approach. The program should first ask the user to input a sentence. Then, the program should scan the sentence for typos, alerting the user if any mistakes are detected. In particular, the program should enforce the following rules:

1. Sentences should begin with a capital letter.
2. Sentences should end with a period.
3. The letter 'q' should always be followed by the letter 'u' (e.g. `qickly` is not allowed, but `quickly` is).
4. Capital letters should only occur at the beginning of a word (e.g. `States` is allowed, but `STates` is not).
5. The pronoun `I` should always be capitalized (e.g. `i ran` is illegal).
6. The letter sequence `cie` should never appear in a word (e.g. `recieved` is illegal, but `received` is allowed).
7. If the letter sequence `ei` appears in a word, the letter before it must be a `c` (e.g. `receive` is okay, but `freind` is not).
8. The number of left parentheses should equal the number of right parentheses. (e.g. `(He ate (slowly)).` is allowed, but `(He ate slowly))` is not.)
9. If a word begins with a vowel, the preceding word must not be "a". If a word begins with a consonant, the preceding word must not be "an".
10. If a word is plural, the next word must not be "is." You can assume that a word is plural if and only if it ends with an 's'.

When an error is detected, the program should print a helpful error message. You may assume that there is at most one error per sentence. Here is what a session might look like:

```
Enter a sentence:
The only thing we have to fear is fear itself.
No typos were detected.

Enter a sentence:
The enemy of my enemy is my freind.
error: 'ei' only allowed after the letter 'c'.

Enter a sentence:
Dude, where's my car?
error: sentence should end with a period.

Enter a sentence:
An good time was had by all.
error: "an" should only appear before words that start with a consonant.
```

Your output format need not match this example exactly. Of course, these spelling and grammar rules are very simplistic. You may ignore the fact that some correct English sentences will be flagged as incorrect by your program. Submit your solution as `checker.c`.

Hint: One approach to solving this problem is to use strings and arrays, but it is possible (and probably easier) to process the sentence character-by-character. Use the `getchar()` function to read in the next character. You may need some temporary variables of type `char` to remember the last few characters that were entered.

4 What to submit

Submit any *two* of the following three files:

1. `retire.c`
2. `intel.c`
3. `checker.c`