California State University, Sacramento
College of Engineering and Computer Science

Computer Science 35: Introduction to Computer Architecture

Spring 2017 – Lab 2 – *Sugar High!*

Overview

Every year you look forward to buying a box of Girl Scout Cookies. Don't lie, you do!

However, **Dr. S** wants you to keep track of your cookie consumption. Cookies contain something called "simple" carbohydrates (aka "carbs") which, in excess, can be quite bad. Your body considers carbs as a type of sugar and, if you have diabetes, this can be detrimental to your health.

So the best way to appreciate your investment (and carb intake) is to write a program that counts the carbs you consume with a snack. Your program will input the number of 3 types of cookies and output the total number of carbs.

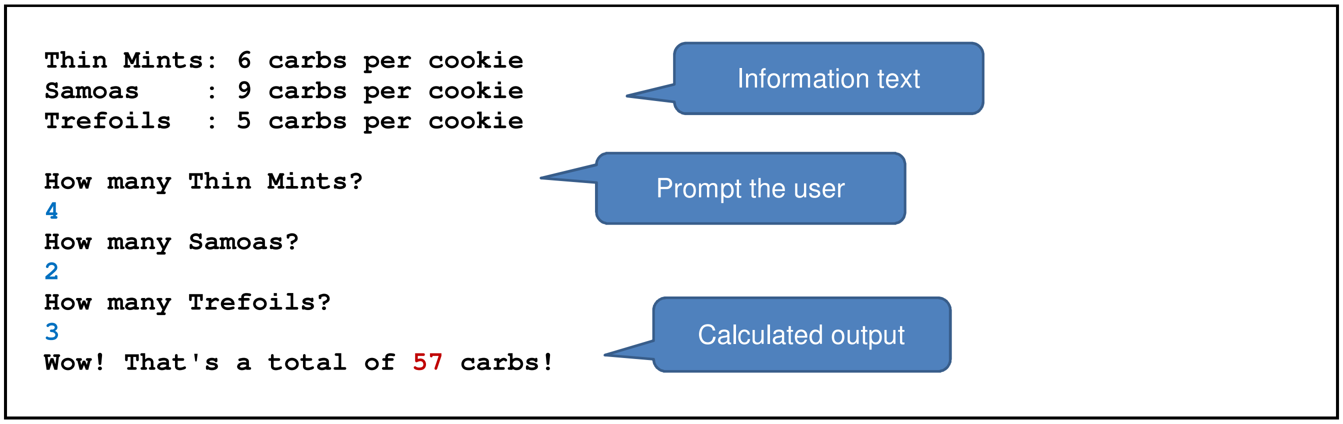
However, not all cookies contain the same number of carbs, so you need to multiply each cookie by the correct number of carbs.

Oh, have some fun too! Create your own cookies types!

Sample Run

The following is a sample run of the program. The user's input is printed in **blue**. The data outputted from your calculations is printed in **red**.

Internally, the program calculated 24 for Thin Mints, 18 for Samoas, and 15 for Trefoils. These are added to a running total resulting in 57!



```
Thin Mints: 6 carbs per cookie
Samoas    : 9 carbs per cookie
Trefoils  : 5 carbs per cookie

How many Thin Mints?
4
How many Samoas?
2
How many Trefoils?
3
Wow! That's a total of 57 carbs!
```

Requirements

You must think of a solution on your own. The requirements are as follows:

1. **Come up with 3 of your own cookie ideas.** Each cookie should have a different amount of carbs.
2. Display a table to the screen (carbs per cookie). *Please see above.*
3. Display a prompt, to the user, for each type of cookie they are going to eat.
4. Input the number of each type of cookie.
5. Calculate the total number of carbs. Tip: use a register to create a running total.
6. Output the total number of carbs with some helpful text.

Hints

- Start off by getting the first multiplication to work and print the correct value.
- Now work on each of the requirements below one at a time. You will turn in the final program, but incremental design is best for labs.

Hello World On x86 Linux

```
# lab1.s
# YOUR NAME HERE
#
# 1. Assemble : as -o lab1.o lab1.s
# 2. Link     : ld -o a.out lab1.o csc35.o
# 3. Execute  : a.out

.data                               #Start the data section
Message:                            #Message is an address
    .ascii "Hello Dr. S!\n\0"       #Create a buffer of ASCII

.text                               #Start the text section
.global _start                      #Make the _start label public

_start:                             #UNIX starts here
    mov  $Message, %rax             #Put the address into rax
    call PrintCString               #Execute the csc35.o subroutine

    call EndProgram                 #Execute the csc35.o subroutine
```

Submitting Your Lab

Run Alpine by typing the following and, then, enter your username and password.

```
alpine
```

Please send an e-mail to yourself (on your Outlook, Google account) to check if Alpine is working. To submit your lab, send the source file (not a.out or the object file) to:

```
dcook@csus.edu
```

UNIX Commands

Editing

Action	Command	Notes
Edit File	`nano` *filename*	"Nano" is an easy to use text editor.
E-Mail	`alpine`	"Alpine" is text-based e-mail application. You will e-mail your assignments it.
Assemble File	`as -o` *objectfile asmfile*	Don't mix up the *objectfile* and *asmfile* fields. It will destroy your program!
Link File	`ld -o` *exefile objectfiles*	Link and create an executable file from one (or more) object files

Folder Navigation

Action	Command	Description
Change current folder	`cd` *foldername*	"Changes Directory"
Go to parent folder	`cd ..`	Think of it as the "back button".
Show current folder	`pwd`	Gives a file path
List files	`ls`	Lists the files in current directory.

File Organization

Action	Command	Description
Create folder	`mkdir` *foldername*	Folders are called directories in UNIX.
Copy file	`cp` *oldfile newfile*	Make a copy of an existing file
Move file	`mv` *filename foldername*	Moves a file to a destination folder
Rename file	`mv` *oldname newname*	Note: same command as "move".
Delete file	`rm` *filename*	Remove (delete) a file. There is **no** undo.

Eat more vegetables!