Lab #2

This lab exercise contributes to your lab mark (3.5%).

1. We reserve the right to ask you to explain and demo your lab solutions.
2. You need to **declare that you wrote the code all by yourself** at the submission webpage.
3. We use the **lab machines in CSC 159** for all marking purposes; make sure that your C programs can be compiled (no warning messages, or any notes) and run on these machines.
4. Download and extract the supplied `lab02.tar` file (http://webdocs.cs.ualberta.ca/~guohui/CMPUT201/Lab02/lab02.tar, or copy from `~guohui/web_docs/CMPUT201/Lab02/lab02.tar`), which contains `ex2q1.c` and `check` needed for the lab.
5. Pack all of your files (your revised `ex2q1.c`, `ex2q2.c`, `ex2q3.c`, `check`, and other files if any) into `submit.tar` by `tar`. You can execute the command `./check submit.tar` to validate that you can extract required files successfully and that they work well (all files in the same directory), where `check` is a program we provide for this lab exercise. Please note that `check` does not necessarily check for correctness of your code.
6. Submit your `submit.tar` to the **designated submission webpage in eClass** as early as possible, and before the due date for your lab section (D01-07 due Sep 21, D08 due Sep 28).

Objectives:

After completing this lab exercise, students should be able to:

- Get the general idea of the layout of a typical small C program
- Design a small single file C program
- Compile and debug C programs

Description:

Part 1: Using `gdb` to trace a C program

When writing a C program, we could make errors that pass the compiler (often referred to as ***bugs***). Your program runs sometimes, but gets into troubles or produces incorrect results at other times. `gdb` is a tool that can be used to trace the program to detect the errors, and one can try to resolve the first error using the tracing output. This is a process called ***debugging***.

The following C program intends to find the maximum value among the a given array of integers, and can pass our standard compilation command `gcc -Wall -std=c99`, yet it has bugs. Try to use the command `gcc -ggdb -Wall -std=c99` to compile a debugging version, and then use `gdb` to locate the first error(s) and fix the program using a minimal set of changes (e.g., do not re-write too many lines of the code).

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

int main() {// This program finds the maximum element in an array
    int length, index = -INT_MAX;
    if (scanf("%d", &length) == 1) {
        int array[length];
        for (int i = 0; i < length; i++) {
            if (scanf("%d", &array[i]) != 1)
                exit(100);
            if (array[index] < array[i])
                index = i;
        }
        printf("The maximum element is %d\n", array[index]);
    }
    return 0;
}
```

Name your final C source code file `ex2q1.c`.

Part 2: Simple Input/Output

We would like a C program to convert CAD currency to USD. United States Dollar (USD) bills come in denominations of $100, $50, $20, $10, $5, $2, and $1. With a conversion rate of $1 CAD = 75 cents USD, determine the least number of USD bills required to make up the Canadian Dollar amount, rounded to the nearest USD (0.50 rounds up). Hint: Use the `%` operator to figure out when to round up.

Your program should check whether the input is valid. If the input is not valid, the program should terminate gracefully with a **non-zero exit code**. The only valid input is a non-negative integer less than $100,000$. Your program's output must match the casing, spelling and formatting of the example outputs below exactly.

Example 1 (note: the user types in "2234" and hits enter here):

```
Enter CAD amount: 2234
Converted USD amount: $1676
$100 bills: 16
 $50 bills: 1
 $20 bills: 1
 $10 bills: 0
  $5 bills: 1
  $2 bills: 0
  $1 bills: 1
```

Example 2 (note: the user types in "78" and hits enter here):

```
Enter CAD amount: 78
Converted USD amount: $59
$100 bills: 0
 $50 bills: 1
 $20 bills: 0
 $10 bills: 0
  $5 bills: 1
  $2 bills: 2
  $1 bills: 0
```

Example 3 (note: the user types in "79" and hits enter here):

```
Enter CAD amount: 79
Converted USD amount: $59
$100 bills: 0
 $50 bills: 1
 $20 bills: 0
 $10 bills: 0
  $5 bills: 1
  $2 bills: 2
  $1 bills: 0
```

Name your final C source code file `ex2q2.c`.

Part 3: Simple Input/Output

An online bookstore has decided to run some promotional discounts this month and they need you to write a C program to automatically apply all their discounts correctly.

The bookstore offers different prices for different types of books.

- $\$15$: hardcover
- $\$12$: softcover
- $\$7$: ebook

It's offering two types of promotional discounts. Coupon discounts are the first type of discounts applied. Each customer can use ***at most one*** of each coupon per purchase; for example, one can't stack two "2 hardcover books" coupons and one "4 hardcover books" to save $\$20$ when buying 5 hardcover books, but one of each to save $\$15$ only.

Percentage discounts are applied all at once, after any coupon discount is applied. For example, if applying a 5% and a 20% percentage discounts, the customer would pay 100% - 5% - 20% = 75% of the discounted total.

> **Percentage discounts:**
> 3% off the final discounted total, if the order includes at least 1 hardcover, 1 softcover, and 1 ebook
> 4% off the final discounted total, if the order includes 3 or more ebooks
> 10% off the final discounted total, on orders over $75
> 15% off the final discounted total, on orders over $150
>
> **Coupon discounts:**
> $5 off on purchases of 2 or more hardcover books
> $10 off on purchases of 4 or more softcover books
> $20 off on purchases of any mix of 6 or more hard or softcover books

The only valid input from the user is a non-negative integer less than $100,000$, specifying the number of books. If the program receives invalid input, terminate gracefully with a **non-zero** exit code. Your answer must be to the nearest cent (hint: there's no need to round the decimals).

Example 1 (note: the user types 79, 23, and 3, for the numbers of books here):

```
How many hardcover books are you buying? 79
How many softcover books are you buying? 23
How many ebooks are you buying? 3
Order total: $983.96
```

Example 2 (note: the user types 3, 3, and 0, for the numbers of books here):

```
How many hardcover books are you buying? 3
How many softcover books are you buying? 3
How many ebooks are you buying? 0
Order total: $56.00
```

Example 3 (note: the user types 1, 1, and 3, for the numbers of books here):

```
How many hardcover books are you buying? 1
How many softcover books are you buying? 1
How many ebooks are you buying? 3
Order total: $44.64
```

Name your final C source code file `ex2q3.c`.

Submitting

1. Run the `check` program (need to change its mode to be executable) and make sure it says you're good to go. Otherwise you won't be able to get full marks

```
#use `tar` to pack your files into `submit.tar`
#then,
chmod 700 check
./check submit.tar
```

2. If you are not working on the lab machines, then scp `submit.tar` back to your local computer; submit it to eClass before the deadline

```
# Run this on your own computer
# Not needed if you are working on a lab machine
scp <ccid>@ug20.cs.ualberta.ca:~/<path_to_your_submit_file>/submit.tar .
```