Lab 4

Lab 4 assignment is due on Friday, March 13, by 9pm in your svn repository. For help, please check the Labs FAQ first. Specifically, please read the Terminal Troubleshooting and SVN Troubleshooting. Also, refer to C Programming Reference and Coding Conventions for C language details.

Please ask all questions about this assignment during the office hours or post questions on piazza.

This lab is to be done on **a Linux workstation or** in **VM** during **any** time of the week. Plan your time accordingly since late submissions will not be accepted.

Introduction to C programming

In this exercise you are going to operate with different conditional and iterative constructs available in C. You will learn about the **while** and the **for** iterative constructs which can be used to repeat a job for a number of times.

Before starting to work on this part, make sure to update your local svn copy. You should have a new directory, called lab4 in your ece120 directory. In this directory you should find file lab4.c which you will need to modify per instructions provided below and to commit the changes to svn. Refer to Lab 3 for instructions on how to compile and execute a program written in C.

Iterative Constructs in C

In every programming language, there are circumstances where we want to do the same thing many times. For instance, you want to print the same words ten times. You could type ten printf function calls, but it is easier to use a loop. The only thing you have to do is to setup a loop that executes the same printf function ten times.

There are three basic types of loop constructs in C:

- "for loop"
- "while loop"
- "do while loop"

We are intentionally going to skip introducing the "do while" loop for now.

The for loop

The "for" loop, loops from one number to another number and increases by a specified value each time.

It uses the following structure:

```
for (start value; end condition; update value)
{
    statements;
    ...
}
```

Here is a simple example of the for loop. This C program prints "Hello World !" 3 times on the screen.

```
#include <stdio.h>
int main()
{
    int i;

    for (i = 0; i < 3; i=i+1)
    {
        printf("Hello World !\n");
    }
    return 0;
}
```

This "for" loop is equivalent to the following infinitely nested if statement:

```
{
    int i = 0;
    if (i < 3)
    {
        printf("Hello World!\n");
        i=i+1;
        if (i < 3)
        {
            printf("Hello World!\n");
            i=i+1;
            if (i < 3)
            {
                printf("Hello World!\n");
                i=i+1
                if (i < 3)
                {
                    ...
                    ...

                    ...
                }
            }
        }
    }
}
```

Let's trace through what happens at each iteration:

In iteration 0

```
i = 0 (first clause of "for" construct)
The program then checks whether i < 3
result = true, hence the program flow goes into the loop
  print Hello World !
The for loop now increments i, hence i = i+1 ( because i=i+1 is the increase value that is mentioned)
therefore i = 0+1
```

In iteration 1

```
(i is now 1)
The program then checks whether i < 3
result = true, hence the program flow goes into the loop
  print Hello World !
The for loop now increments i, hence i = i+1 ( because i=i+1 is the increase value that is mentioned)
therefore i = 1+1
```

In iteration 2

```
(i is now 2)
The program then checks whether i < 3
result = true, hence the program flow goes into the loop
  print Hello World !
The for loop now increments i, hence i = i+1 ( because i=i+1 is the increase value that is mentioned)
therefore i = 2+1
```

In iteration 3

```
(i is now 3)
The program then checks whether i < 3
result = false, hence the program flow breaks out of the loop
```

Final output

```
Hello World !
Hello World !
Hello World !
```

for **(index = 0; index < max_index; index=index+1)**
{
...
}

The above usage of "for" is a time honored way of iterating through integers in the interval [0, max_index) (0 <= index < max_index). It's actually quite elegant because the loop executes exactly "max_index" times (with "index" taking values from 0 to max_index - 1). You will see it quite often in C code and are encouraged to use it yourself.

The while loop

The while loop does the same task as the for loop but is used when you don't know how many times a particular task has to be done.

Here is a simple example of the while loop. In this C program, an integer value is entered by the user. This value then tells our program the number of times "Hello World !" has to be printed on the screen. This program effectively does the same task as the above program if the user enters 3

```
#include <stdio.h>
int main()
{
    int count = 0, number_of_times;
    scanf("%d", &number_of_times);

    while(count < number_of_times)
    {
        printf("Hello World !\n");
        count=count+1;
    }

    return 0;
}
```

As you can see for simple looping operations such as shown above, using the "for" loop results in a much more concise and cleaner code. Hence, we suggest you use the "for" loop (instead of the "while" loop) to carry out simple tasks like this. Below, we show a much more appropriate use of the while loop.

A more appropriate use of the while loop

The code shown below is used to accept a series of characters from a user. When the user hits return (enter on the keyboard), the program stops accepting characters and displays how many characters the user has entered. In a real program you would store these characters in an array for further manipulation. We will introduce arrays and operations on them in ECE 220.

```
#include <stdio.h>
int main()
{
    char c;
    int count = 0;

    while (c != '\n')
    {
        c = getchar();
        count=count+1;
    }

    printf("\n You have entered %d characters\n", count-1);
    return 0;
}
```

The rest of this lab's assignment

Here you are given a program stored in lab4.c file in which the iteration is done using a "for" loop.

```
#include <stdio.h>
#include <math.h>

int main()
{
    int a;

    for (a = 256; a > 1; a = sqrt(a))
    {
        printf("%d\n", a);
    }
    printf("%d\n", 1);

    return 0;
}
```

Your task is to convert the above program and use the "while" loop construct instead of the "for" loop construct.

Edit this code in lab4.c to use only the "while" loop and upload the modified C file to your SVN repository. That's it for this lab!

You will probably see the error **undefined reference to `sqrt'** when you try to compile lab4.c. The problem is that the sqrt() function is defined in a C math library (with header file <math.h>), not in the standard C library (with header file <stdio.h>). This C math library provides other functions such as pow(), log(), sin(), cos(), tan(), etc.

But the compiler does not automatically know that it should link the math library during the compilation process. To use a math function provided in the <math.h> header file, you must give the **-lm** flag to the **gcc** compiler like so:

```
gcc -lm -g -std=c99 -Wall -Werror -o lab4 lab4.c
```

When rewriting the "for" loop into a "while" loop, make sure that the variable "a" is initially assigned the value 256.

Remember, the output of the series should be exactly the same as the output of the program given using "for" loop.

HINT: Try unrolling the "for" loop as shown above. This will give you a clear idea of what is happening and will help you transform your code to use a "while" loop.