CS2211a Lab No. 5
Introduction to C

Tuesday October 14, 2014 (sections 3 and 2),
Wednesday October 15, 2014 (sections 6 & 7), and
Thursday October 16, 2014 (sections 4 and 5)

Location: ***MC10*** lab

The objective of this lab is:

- To practice the C compilation process
- To practice C formatted input/output, as well as C expressions and selection statements

If you would like to leave, and at least 30 minutes have passed, raise your hand and wait for the TA.
Show the TA what you did. If, and only if, you did a reasonable effort during the lab, he/she will give you the lab mark.

==========

1. Type in (using an editor) and then compile (using **gcc**) the following `hello.c` program. If you do not have any compilation errors, the compiler will generate an executable program called `a.out`. To execute this program, simply type `a.out` and hit enter.

```
#include <stdio.h>
int main(void)
{ printf(("Hello world\n");
}
```

2. If you want to name the generated executable file with something other than `a.out`, you should use **gcc** with **-o** option, e.g., `gcc –o hello hello.c` (in this case, you should type `hello` to execute the program)

3. By default, **gcc** compiles programs using **C89**. To compile using **C99**, you should use "**gcc -std=c99**". Recompile `hello.c` program using **C99** instead of **C89**. Do you get a warning message from the compiler? If so, what needs to be done to make it go away?

4. It is a good practice to use "**gcc -Wall**" to check all warnings in your program. Try again to compile the *original* `hello.c` program with "**gcc -Wall**" using **C89** and **C99**.

5. You can force **gcc** to stop after the preprocessing stage by using **-E** option. Type in the following `simple.c` program. Execute the following two commands:

```
gcc –E simple.c
gcc –E –std=c99 simple.c
```

What is the difference between the two outputs?

```
#define MIN 0
#define MAX 100
int main(void)
{ int i, j, m, n;
  i = MAX;
  /* First command line
  */
  j = MIN;
  // Second command line
  m = i + j;
  n = MAX + MIN;
  return 0;
}
```

6. Write a program that declares several `int` and `float` variables (without initializing them) and then prints their values. Is there any pattern in the values?

7. What will happen when you attempt to print an arithmetic expression using the *wrong* conversion specifier, i.e., printing `int` expression using `%f` or `float` expression using `%d`?

8. Mentally evaluate the following `printf` function calls and write your answer on a piece of paper.
   Write a program that prints the values of the output. Verify your answers with the program answers

```
(a)  printf("*%20.10d*\n*%-20.10d*\n*%.10d*\n*%-20d*\n",
             123456,123456,-123456,-123456);

(b)  printf("%.4f\n", 83.162);

(c)  printf("%12.5e\n", 30.253);

(d)  printf("%-6.2g\n", .0000009979);
```

9. Mentally evaluate the following expressions and write the answer on a piece of paper.
   Write a program that prints the value of these expressions. Verify your answers with the program answers.
   (a) `13 % 4`, `-13 % 4`, `13 % -4`, and `-13 % -4`
   (b) `13 / 4`, `-13 / 4`, `13 / -4`, and `-13 / -4`

10. Show the output produced by the following program fragment.

```
int i = 1, j = 2;
int k = 3, m = 4;
i %= j++ % (k += --m);
printf("%d %d %d %d\n", i++, ++j, k--, --m);
```

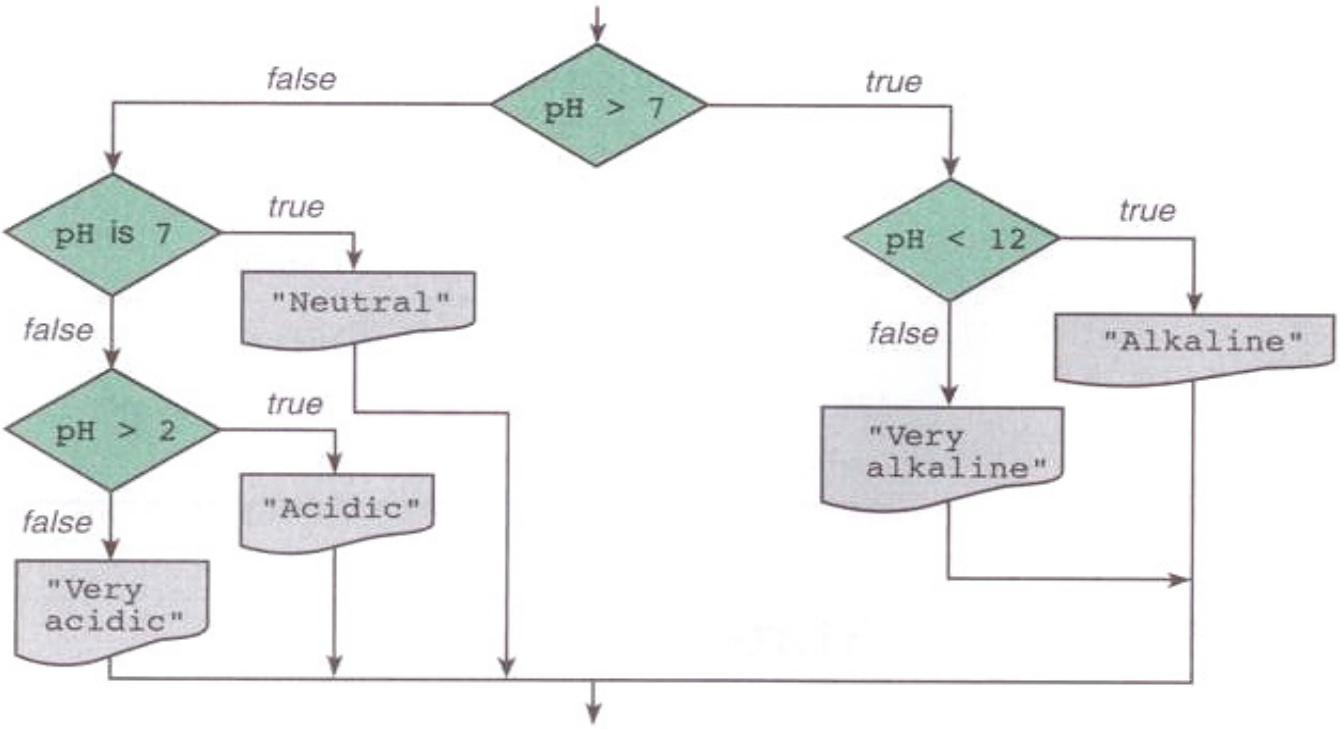
11. Show the output produced by the following program fragment.

```
int i = 1, j = 2;
int k = 3, m = 4;
i *= j / - (k -= ++m);
printf("%d %d %d %d\n", i++, ++j, k--, --m);
```

12. De Morgan's laws state that
    the expression `!(condition1 && condition2)` is logically equivalent to
    the expression `(!condition1 || !condition2)`.
    In addition,
    the expression `!(condition1 || condition2)` is logically equivalent to
    the expression `(!condition1 && !condition2)`.
    Use De Morgan's laws to write equivalent expressions for each of the following, and then write a program to show that both the original expression and the new expression in each case are equivalent.

```
(a)    !(x <  5) && !(y >= 7)
(b)    !(a == b) || !(g != 5)
(c)   !((x <= 8) &&  (y >  4))
(d)   !((i >  4) ||  (j <= 6))
```

13. Write a program to implement the following flowchart *segment*. The program should get the value of pH from the user. Test your program to make sure that it produces correct answers.



14. Rewrite the program in Q13 using only ONE **`printf()`** function call and ONE ***compound conditional expression.***

15. Rewrite the following program fragment by replacing the `switch` statement with
    (a) Nested `if .. else` statement
    (b) A series of `if` statements without `else`

    Be careful to deal with the `default` case properly.

```
for(int i=1; i<=8; i++)
{
 switch(i)
  {
    case 1:
    case 2:  printf("%d ==> a\n", i);
             break;

    case 3:
    case 4:
    case 5:  printf("%d ==> b\n", i);
             break;

    case 6:  printf("%d ==> c\n", i);
             break;

    default: printf("%d ==> d\n", i);
  }
}
```