BIRLA INSTTUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)
CS F111 Computer Programming
LAB SESSION #3
(Writing basic C Programs)

We will start implementing and executing C programs from today's lab. All labs, henceforth, shall focus on programming in C. We expect you to clearly understand how to create files using sublime text and access/edit them on the UNIX terminal, either in WSL or in Ubuntu. Please follow the video instructions of lab session 2 if you are not familiar with it.

Create a dedicated directory "**myprogs**" in your home directory if you are an ubuntu user or in '**D**' (or '**E**' or any) drive of windows if you are a WSL user. You can create/edit your C program-related files using sublime text and store them in "**myprogs**" directory. You can create a directory ***inside*** "**myprogs**" for this particular lab, say a directory with the name "**lab3**".

We will write our first C Program in this lab. Please revise the lecture slides of module 3 before you start. The explanation includes what are the header files, **`printf()`**, **`scanf()`**, etc. We will use them to write our programs in this lab. Let us begin.

1. Write a program that prints "Hello World!".

In this program we will use the **`printf()`** statement to print "Hello World!". Here is the program:

```
#include <stdio.h>

int main()
{
      printf("Hello World!");
      return 0;
}
```

Create a file ***hello_world.c*** in the directory "**lab3**" using the sublime text editor and copy this program into it and save the file. Please note that any C program must be stored in a file that has ".c" extension. This is required by the C compiler. C compiler also reads files with ".h" extension. We will study about them later.

On your terminal, navigate to "**lab3**" directory (using the **cd** command). Then run the following commands:

```
$ gcc hello_world.c
$ ./a.out
```

You will see that "Hello World!" is printed on the screen. However, the terminal prompt has appeared beside the printed statement. To move the prompt to the next line, you shall need to replace the previous **`printf`** statement with the following new statement:

```
      printf("Hello World!\n");
```

"**`\n`**" is the new line character that is used to move to the next line on the terminal. It is one of the many **escape sequences** that the C language offers. Check out this link to know more about escape sequences.

Please note a few things in the program we just wrote. **`int main()`** is the definition of the main function, which is the place from where any C program starts its execution.

`return 0` is the statement that states that the program has terminated. The return value is the exit code of your program. The shell (or any other application that ran the program) can read and use it. The **0** exit code is a widely accepted convention for '**OK the program**

execution was successful'. Any arbitrary C Program you write always starts with **`int main()`** and should contain **`return`** statement at the end.

Also, when you compiled your program with **`gcc hello_world.c`** command, it creates an executable with the name **`a.out`**. You can use **`ls`** command after compilation to check if the executable is actually created. And then you execute that executable using the command **`./a.out`**. This executes the program and performs the intended task of the program, which in our case is printing of "Hello World!" statement.

Okay, let us now advance further into our second program for today. In this program we will see how to add two numbers and print their sum. Here it is:

2. Write a program to add two numbers and print its sum.

In this program, we will make use of *integer variables* to store the numbers. A variable is nothing but a placeholder for a value that you want to store. Let us see the following program:

```
#include <stdio.h>

int main()
{
      int num1, num2, num3;
      num1 = 2;
      num2 = 4;
      num3 = num1 + num2; // computing the sum of num1 and num2
      printf("The sum is: %d \n", num3); // printing the sum
      return 0;
}
```

Create a file ***sum.c*** in the directory "**lab3**" using sublime text editor and copy this program into it and save the file. Then compile it using **`gcc sum.c`** and execute it with **`./a.out`**.

In this program we had used the variables **`num1`**, **`num2`** and **`num3`**, which are of the type **`int`** (or integer). **`int`** variables in C can hold both negative and positive integers. In this program, we store the value 2 in the variable **`num1`**, value 4 in the variable **`num2`**, compute the sum of **`num1`** and **`num3`** store in the variable **`num3`**.

Then we print the value of **`num3`** using the **`printf`** statement. Please note that "**`%d`**" in the above **`printf`** statement indicates that the value we are trying to print is of the type "**int**" (or integer). This is known as a *format specifier*. We shall have to use "**`%f`**" for printing float values, "**`%lf`**" for printing double values and "**`%c`**" to print characters, which we will see in further lecture and lab sessions.

3. What was there in **num1, num2 & num3** before you assigned any value in the program 2? Are there some default values? Try printing the values before the assignment.

Alright, we have seen how to compute sum of two numbers and print its value. Now, the user wishes to give the values of **`num1`** and **`num2`** during the program execution, instead of giving their values while writing the program. For this we can use the **`scanf`** statement that scans for input from the user during the program execution and stores the value in a variable. Let us see it in the next program:

4. Take input of two numbers from the user and print their sum.

Here is the program:

```
#include <stdio.h>

int main()
{
      int num1, num2, num3;
      printf("Please enter the first number:");
      scanf("%d", &num1); // captures an integer value and stores in num1
      printf("Please enter the second number:");
      scanf("%d", &num2); // captures an integer value and stores in num2
      num3 = num1 + num2; // computing the sum of num1 and num2
      printf("The sum is: %d \n", num3); // printing the sum
      return 0;
}
```

Create a file ***sum2.c*** in the directory "**lab3**" using sublime text editor and copy this program into it and save the file. Then compile it using **`gcc sum2.c`** and execute it with **`./a.out`**. Please note that each time you create compile a new file, the old ***a.out*** is replaced with a new ***a.out*** specific to the most recently compiled program. If you want to keep a separate executable for each program, then follow the following commands:

```
$ gcc sum2.c -o sum2_exe
$ ./sum2_exe
```

The executable that got created after compilation of ***sum2.c*** is now ***sum2_exe*** instead of ***a.out***. This is what the -o flag is used for. And you can execute this by **`./sum2_exe`**.

The above program, takes two values from the user, prints its sum. Note the syntax of the **`scanf()`** statement:

```
scanf("%d",&num1);
```

This indicates that the value you are scanning is an integer (**`%d`**) and you want to store that value in **`num1`** by specifying the address of **`num1`**. Address is nothing but the location in the main memory where **`num1`** is stored. The address of a variable is accessed by prefixing & symbol before the variable name. In this case we had used **`&num1`** to denote the address of the variable **`num1`** in the main memory, which you had passed as input to the **`scanf`** statement for it to store the scanned value into **`num1`**. This is how **`scanf`** works.

Now, let us try to write a few programs by ourselves using simple **`printf`**, **`scanf`** statements, integer variables and operations over integer variables. A few operations over integer variables include:

Addition:
```
num3 = num1 + num2
```

Subtraction:
```
num3 = num1 - num2
```

Multiplication:
```
num3 = num1 * num2
```

Integer Division:
```
num3 = num1 / num2   (NOTE: This is integer division 3/2 is not 1.5)
```

Modulus (or remainder after division)
```
num3 = num1 % num2
```

5. Write a C program to calculate the total distance travelled by a vehicle in "t" seconds, given by: $d = ut + \frac{1}{2} at^2$. Get user input for u, a and t. Output the value of d. You can create intermediate variables to store intermediate values. For example you can create a temporary variable **temp1** to store the multiplication of u and t, and another temporary variable **temp2** to store $\frac{1}{2} at^2$. And then add **temp1** and **temp2** to get the value of d.

6. Take the above C program and copy it without the .c extension (Hint: use the cp command on Linux). Now try compiling this copy. Some C compilers will complain; others will not. On Unix systems, the complaint may be quite cryptic. What happens on your system?

7. Type the following code in your system and observe the format of the output printed by various printf statement in the program

```
#include <stdio.h>
int main()
{
   char ch = 'A';    char str[20] = "Computer";
   float flt = 10.234;    int no = 150;   double dbl = 20.123456;
   printf("Character is %c \n", ch);
   printf("String is %s \n" , str);
   printf("Float value is %f \n", flt);
   printf("Integer value is %d\n" , no);
   printf("Double value is %lf \n", dbl);
   printf("Octal value is %o \n", no);
   printf("Hexadecimal value is %x \n", no);
   return 0;
}
```

   Please note various types of format specifiers used to print various types of values.

8. Make the following changes in the code written in question 4 and observe the effect of each change on the compilation process:

   a) Delete a semicolon and see what happens.
   b) Leave out one of the braces
   c) Remove one of the parentheses next to the main function

By simulating errors like these, you can learn about different compiler errors, and that will make your typos easier to find when you make them for real!

Additional Practice Exercises for Home Work

1. Write a C program that reads in an integer denoting number of days. It prints the number of years, number of months and the number of days that constitute the input number of days. For example, if the input number is 403, it should print 1(year), 1(month), 13(days). For simplicity: there is no need to consider leap years and assume all months have 30 days. [**Hint:** Use modulus (`%`) and division (`/`) operators. **End of Hint**]

2. Write a C program to enter Principal (P), Time (T), Rate of Interest (R) and calculate Simple Interest (SI) and show it to the user. Note that SI = PxRxT.

3. Write a C program to print a block F using hash (#), where the F has a height of six characters and width of five and four characters. And also to print a big 'C'.
Expected Output:

```
######
#
#
#####
#
#
#
  ######
 ##    ##
#
#
#
#
#
 ##    ##
  ######
```

4. There are four main stages through which a source code is passed in order to finally get a runnable executable. They are: *pre-processing*, *compilation*, *assembly* and *linking*. By invoking **`gcc`** command, all these steps are accomplished, and you get the resultant executable in ***a.out***. You can stop this pipeline at a specific stage by giving specific options for **`gcc`**.
For instance, try **`gcc -S lab3_1.c`** for just invoking the assembler. The assembly code for the C program is now generated and stored in ***lab3_1.s*** that you can inspect using sublime text. You may not understand the statements, but you should learn to identify how an assembly program looks like.

5. This exercise is meant for you to explore and learn more about **`printf()`** format specifiers. Try each of these and infer what is the meaning:

int x = 12; printf("%5d", x);	int x=12; printf("%-5d", x);	float x = 234.5678; printf("%-8.2f", x);
float x = 234.5678; printf("%+8.2f",x);	float x = 234.5678; printf("%+-8.2f",x);	char ch = 'Y'; printf("%c", ch);

6. A computer manufacturing company has the following monthly compensation policy to their salespersons:
Minimum base salary : 1500.00
Bonus for every computer sold : 200.00
Commission on the total monthly sales : 2 per cent

Since the prices of computers are changing, the sale price of each computer is fixed at the beginning of every month. Write a C program, "**computer_sales.c**", to compute a sales-person's bonus, commission and gross salary. Your program should take the number of computers sold (in a month) and the sale price of a computer as user input.