OBJECTIVES

1. Learn to handle simple user input/output using basic C functions.

GENERAL INFORMATION

All useful programs require some sort of input data and generate output. Input data can be read from non-volatile or volatile storage, network data streams from hardware interfaces, or simply from the user terminal (keyboard). These sources can also be output destinations, simplest one being the terminal again (computer monitor). We are to practice on and learn basic usage of printf and scanf standard library functions in this experiment.

We have already seen the usage of printf function in previous lab classes with MATLAB version of it, fprintf. C version is not very different. The syntax (prototype) is

```c
int printf(char *format,...);
```

where format string usually contains " " (double quote) enclosed formatting sequences that start with %. All other characters are printed out as is. Special characters are represented by escape sequences.

- `%d` : decimal (integer)
- `%ld` : decimal (long)
- `%f` : float
- `%lf` : double
- `%n.mf` (n, m are integers): m digits after radix, total n digits (if possible)
- `%s` : strings
- `%c` : single character

Escape Characters are

- `
` : next line
- `	` : tab

Example call:

```c
printf("Var=%7.1f and x is %d\n",v,x);
```

scanf function use the same formatting sequences except that since it is used to read input from the interfaces you should not use normal text within the format string. That is, use only % sequences. scanf reads the input (keyboard buffer) according to the format string and stores them into addresses given as arguments.

```c
int A; float X;
scanf("%d %f",&A,&X);
```

reads a decimal integer and a floating point number (also decimal) from the buffer (after user presses Enter) and stores them into A and X respectively. The & signs in front of A and X mean "address of". &A means "address of variable A". scanf requires addresses of variables, not their names. Use them as described here, and we will learn addresses and pointers later in the class.

EXPERIMENT

1. Write a program (with main() function only) that asks the user to enter a, b and c coefficients of a 2nd degree polynomial and calculates its real roots if exist. Program prints them out afterwards. Do not forget to #include <math.h> for sqrt function. Also remember what we did in the class when the console window closes immediately after the program termination before we see the results.

```
Enter a, b, c : 2.1, 4.6, -11.5
Roots : X1=1.488507  X2=-3.678983
```

What happens when you omit commas and use only spaces to separate input numbers? What happens when you use multiple spaces between input numbers?

2. In order to detect complex roots and display "roots are complex" message, use if keyword. The syntax is not much different from MATLAB if.

```c
if(condition) { statements } /* if(delta<0) { printf("blah blah... */
```

You should also use else here; if( ){}else{} (may span multiple lines)

How would you calculate and display complex roots? Explain.

QUESTIONS : already asked above