BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)

Second Semester 2017-18 CS F111 Computer Programming

LABORATORY SESSION #5 SOLUTIONS

(Types, Operators and Expressions)

1. Conclusion – data type `char` is implemented as signed by default.

2. In the statement `unsigned long long val = -1;` there is a type conversion that takes place, and `val` now has all ones stored internally. Since `long long int` has the largest memory allocation of an integral data type in any system, this stores the maximum positive integer that can be stored.

For the second part of the question, please read that part of the `printf()` manual which lists the conversion specifiers (and length modifiers). You should become familiar with the at least the following: `%c`, `%d`, `%f` and `%lf` as `printf()` and `scanf()` arguments for char, int, float and double, respectively; `%u`, `%o` and `%x` for unsigned, octal and hexadecimal; `%hd`, `%ld`, `%lld`, `%Lf` for `short int`, `long int`, `long long int` and `long double`. Students should record these in their lab notebooks.

3. Here is the output of the program:

```
Sizeof (char) = 1 bytes
Sizeof (short)= 2 bytes
Sizeof (int)= 4 bytes
Sizeof (long)= 8 bytes
Sizeof (float)= 4 bytes
Sizeof (double)= 8 bytes
Sizeof (1.55)= 8 bytes
Sizeof (1.55L)= 16 bytes
Sizeof (str)= 6 bytes
```

4. Shown is the output alongside the code. In both instances, the right side of the logical operator is not evaluated (short-circuited).

```
int a=0, b=0, x;
x = 0 && (a=b=777);
printf("%d %d %d\n", a, b, x);  /* Answer: 0  0  0 */
x = 777 || (a = ++b);
printf("%d %d %d\n", a, b, x);  /* Answer: 0  0  1 */
```

5. You should learn that ASCII has only positive numbers mapped to characters. The program itself is a simple 3-line code:

```
int main()
{
int i; /* or unsigned char i */
for (i = 0; i <= 127; ++i)
  printf("%d corresponds to: **%c**\n", i, i);
}
```

Please note that the decimal values 0 through 31, and 127, represent non-printable control characters. All other characters can be printed by the computer, i.e. displayed on the screen or printed on printers, and are called *printable characters.*

6. Here is the complete C program:

```
#include<stdio.h>
#include<math.h>
int main()
        {
          double  a,b,c,x1,x2;
          printf("Enter values of a: ");
          scanf("%lf",&a);
          printf("Enter values of b: ");
          scanf("%lf",&b);
          printf("Enter values of c: ");
          scanf("%lf",&c);

          x1 = (-b + sqrt((b*b) - (4*a*c)))/(2*a);
          x2 = (-b - sqrt((b*b) - (4*a*c)))/(2*a);

          printf("The first root of the quadratic eqn is %lf\n", x1);
          printf("The second root of the quadratic eqn is %lf\n", x2);
          return 0;
        }
```

(i) x1 = –0.666667, x2 = –1.000000

(ii) x1 = 1.000000, x2 = 1.000000

(iii) x1 = –nan, x2 = –nan (imaginary roots, and hence not a number (nan))

7. The goal of this exercise is for you to understand that floating-point numbers are stored as approximate values in their IEEE representations. The blog post explains it well.