Little Endian Vs Big Endian
Little Endian means that the lower order byte of the number is stored in memory at the lowest address, and the higher order byte is stored at the highest address. That is, the little end comes first.

For example, a 4 byte, 32-bit integer
Byte3 Byte2 Byte1 Byte0

will be arranged in memory as follows:
Base_Address+0 Byte0
Base_Address+1 Byte1
Base_Address+2 Byte2
Base_Address+3 Byte3

Example : Intel processors use "Little Endian" byte order.

"Big Endian" means that the higher order byte of the number is stored in memory at the lowest address, and the lower order byte at the highest address. The big end comes first.

Base_Address+0 Byte3
Base_Address+1 Byte2
Base_Address+2 Byte1
Base_Address+3 Byte0

Example : Motorola, Solaris processors use "Big Endian" byte order.

Assignment 1:
Write a C program to swap the content of 2 variables entered through the command line using function and pointer.

Assignment 2:
Write a C program to assign values to each members of the following structure. Pass the populated structure to a function Using call-by-value and another function using call-by-address and print the value of each member of the structure.

```
struct student_info{
int roll_no;
char name[50];
float CGPA;
struct dob age;
};
struct dob{
int date;
int month;
int year;
};
```

Assignment 3:
Write a C program to extract each byte from a given number and store them in separate character variables and print the content of those variables.

Assignment 4:
Write a C program to check whether the Host machine is in Little Endian or Big Endian. Enter a number, print the content of each byte location and Convert the Endianness of the same i.e. Little to

Big Endian and vice-versa.

Assignment 5:
Write a C Program to enter a number and store the number across the following structure and print the content of each member of the structure. Then aggregate each member of the structure to form the original number and print the same.

```
struct pkt{
char ch1;
char ch2[2];
char ch3;
};
```

Assignment 6:
Write a C program to populate the following structure which will keep 16 bytes of data. Extract 2 bytes of data(using another structure pkt1) from the below struct at a single point of time and print the same. Continue the same till the end of data in the structure.

```
struct pkt{
        char ch[8];
        int num1;
        int num2;
};
struct pkt1{
        char ch1;
        char ch2;
};
```