Q1. Swapping through cmd line args

#include<stdio.h>

void swap(char \*a,char \*b)

{

    char tmp=\*a;

    \*a=\*b;

    \*b=tmp;

}

int main(int arg, char \*argv[])

{

    char a,b;

    a=\*argv[1];

    b=\*argv[2];

    printf("a= %c\nb= %c\n",a,b);

    swap(&a,&b);

    printf("\nAfter swapping...\na= %c\nb= %c",a,b);

}

Q2. Student info struct call by value and call by address

#include<stdio.h>

struct dob{

    int day,month,year;

};

typedef struct student\_info{

    int roll\_no;

    char name[50];

    float CGPA;

    struct dob age;

}INFO;

void call\_value(INFO ob){

    printf("The Students info are---\n");

    printf("roll no: %d\n",ob.roll\_no);

    printf("name: %s\n",ob.name);

    printf("cgpa: %.2f\n",ob.CGPA);

    printf("dob: %d %d %d\n",ob.age.day,ob.age.month, ob.age.year);

}

void call\_address(INFO \*ob){

    printf("The Students info are---\n");

    printf("roll no: %d\n",ob->roll\_no);

    printf("name: %s\n",ob->name);

    printf("cgpa: %.2f\n",ob->CGPA);

    printf("dob: %d %d %d\n",ob->age.day,ob->age.month,ob->age.year);

}

int main()

{

    INFO ob[2];

    for(int i=0;i<2;i++){

        printf("Enter Student %d info: \n",i+1);

        printf("Name: ");

        scanf("%s",&ob[i].name);

        printf("roll: ");

        scanf("%d",&ob[i].roll\_no);

        printf("cgpa: ");

        scanf("%f",&ob[i].CGPA);

        printf("dob day: ");

        scanf("%d",&ob[i].age.day);

        printf("dob month: ");

        scanf("%d",&ob[i].age.month);

        printf("dob year: ");

        scanf("%d",&ob[i].age.year);

    }

    printf("\n\nStudent details called by value...\n");

    call\_value(ob[0]);

    printf("\n\nStudent details called by address...\n");

    call\_address(&ob[1]);

}

Q3

#include <stdio.h>

int main() {

    unsigned int number;

    printf("The input number= ");

    scanf("%u", &number);

    // Extract each byte and convert them from binary to decimal (base 10)

    unsigned char byte1 = (number >> 24) & 0xFF;

    unsigned char byte2 = (number >> 16) & 0xFF;

    unsigned char byte3 = (number >> 8) & 0xFF;

    unsigned char byte4 = number & 0xFF;

    printf("Byte 1: %u\n", byte1);

    printf("Byte 2: %u\n", byte2);

    printf("Byte 3: %u\n", byte3);

    printf("Byte 4 : %u\n", byte4);

}

Q4.

#include <stdio.h>

typedef struct pkt {

    char ch1;

    char ch2[2];

    char ch3;

}PKT;

int main() {

    unsigned int originalNumber;

    PKT packet;

    printf("The input numbe= ");

    scanf("%u", &originalNumber);

    packet.ch1 = originalNumber & 0xFF;

    packet.ch2[0] = (originalNumber >> 8) & 0xFF;

    packet.ch2[1] = (originalNumber >> 16) & 0xFF;

    packet.ch3 = (originalNumber >> 24) & 0xFF;

    printf("Digit in byte 1= %u\n",packet.ch1);

    printf("Digit in byte 2= %u\n",packet.ch2[0]);

    printf("Digit in byte 3= %u\n",packet.ch2[1]);

    printf("Digit in byte 4= %u\n",packet.ch3);

    printf("Content of the structure:\n");

    printf("ch1: %u\n", packet.ch1);

    printf("ch2: %u %u\n", packet.ch2[0], packet.ch2[1]);

    printf("ch3: %u\n", packet.ch3);

    unsigned int reconstructedNumber =

        ((unsigned int)packet.ch3 << 24) +

        ((unsigned int)packet.ch2[1] << 16) +

        ((unsigned int)packet.ch2[0] << 8) +

        (unsigned int)packet.ch1;

    printf("Reconstructed Number: %u\n", reconstructedNumber);

}

Q5.

#include <stdio.h>

int isLittleEndian()

{

    int num = 1;

    char \*ptr = (char \*)&num;

    return (\*ptr == 1);

}

unsigned int swapEndianness(unsigned int number)

{

    return ((number >> 24) & 0xFF) |

           ((number >> 8) & 0xFF00) |

           ((number << 8) & 0xFF0000) |

           ((number << 24) & 0xFF000000);

}

int main()

{

    unsigned int number;

    printf("Enter a number: ");

    scanf("%u", &number);

    printf("\n------------------------------------------\n");

    printf("\nMemory Address\t->\tValue\n");

    printf("\n------------------------------------------\n");

    printf("%u\t\t  \t%u\n",&((char\*)&number)[0],(int)(number & 0xFF));

    printf("%u\t\t  \t%u\n",&((char\*)&number)[1],(int)((number >> 8) & 0xFF));

    printf("%u\t\t  \t%u\n",&((char\*)&number)[2],(int)((number >> 16) & 0xFF));

    printf("%u\t\t  \t%u\n",&((char\*)&number)[3],(int)((number >> 24) & 0xFF));

    if (isLittleEndian())

    {

        printf("\nThe LSB of the number is stored at the lowest memory address\nHence, the host machine is Little Endian.\n");

    }

    else

    {

        printf("\nThe LSB of the number is stored at the largest memory address\nHence, the host machine is Big Endian.\n\n");

    }

    printf("The Number is converted to Big Endian now\nMemory Representation of the Number...");

    unsigned int convertedNumber = swapEndianness(number);

    printf("\n------------------------------------------\n");

    printf("\nMemory Address\t->\tValue\n");

    printf("\n------------------------------------------\n");

    printf("%u\t\t  \t%u\n",&((char\*)&convertedNumber)[0],(int)(convertedNumber & 0xFF));

    printf("%u\t\t  \t%u\n",&((char\*)&convertedNumber)[1],(int)((convertedNumber >> 8) & 0xFF));

    printf("%u\t\t  \t%u\n",&((char\*)&convertedNumber)[2],(int)((convertedNumber >> 16) & 0xFF));

    printf("%u\t\t  \t%u\n",&((char\*)&convertedNumber)[3],(int)((convertedNumber >> 24) & 0xFF));

    printf("\n\nThe number after Endianness conversion: %d",convertedNumber);

}