**Q1.** Write a C program to swap the content of 2 variables entered through the command

line using function and pointer.

**CODE:**

#include<stdio.h>

void swap(int \*a,int \*b){

int temp;

temp=\*b;

\*b=\*a;

\*a=temp;

}

int main(int argc, char \*argv[]){

int c=atoi(argv[1]);

int d=atoi(argv[2]);

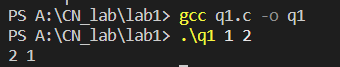
swap(&c,&d);

printf("%d %d",c,d);

return 0;

}

**OUTPUT:**



**Q2.** Write a C program to assign values to each member 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;

};

**CODE:**

#include<stdio.h>

struct dob{

int day;

int month;

int year;

};

struct student\_info{

int roll;

char name[50];

float CGPA;

struct dob age;

};

void fun(struct student\_info s){

printf("Name: ");

puts(s.name);

printf("Roll: %d\n",s.roll);

printf("CGPA: %0.2f\n",s.CGPA);

printf("Date of birth: %d-%d-%d\n",s.age.day,s.age.month,s.age.year);

}

void fun1(struct student\_info \*s){

printf("\n\nName: ");

puts(s->name);

printf("Roll: %d\n",s->roll);

printf("CGPA: %0.2f\n",s->CGPA);

printf("Date of birth: %d-%d-%d\n",s->age.day,s->age.month,s->age.year);

}

int main(){

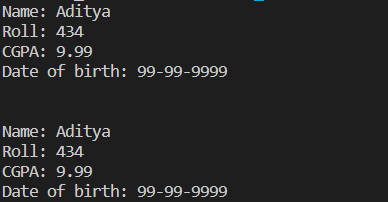
struct student\_info s={434,"Aditya",9.99,{99,99,9999}};

fun(s);

fun1(&s);

}

OUTPUT:



**Q3.** 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.

**CODE:**

#include <stdio.h>

int main() {

unsigned int number;

printf("Enter a number (unsigned integer): ");

scanf("%u", &number);

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", byte4);

printf("Byte 2: %u\n", byte3);

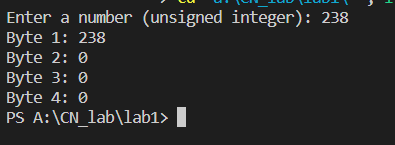
printf("Byte 3: %u\n", byte2);

printf("Byte 4: %u\n", byte1);

return 0;

}

**OUTPUT:**



**Q4.**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;

};

**CODE:**

#include <stdio.h>

struct pkt {

char ch1;

char ch2[2];

char ch3;

};

int main() {

unsigned int number;

printf("Enter a number (unsigned integer): ");

scanf("%u", &number);

struct pkt packet;

packet.ch1 = (number >> 24) & 0xFF;

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

packet.ch2[1] = (number >> 8) & 0xFF;

packet.ch3 = number & 0xFF;

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

printf("ch1: %d\n", packet.ch1);

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

printf("ch3: %d\n", packet.ch3);

unsigned int originalNumber = 0;

originalNumber |= ((unsigned int)packet.ch1) << 24;

originalNumber |= ((unsigned int)packet.ch2[0]) << 16;

originalNumber |= ((unsigned int)packet.ch2[1]) << 8;

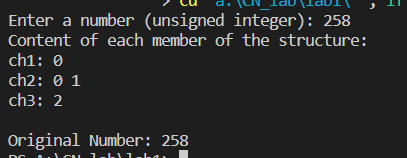
originalNumber |= packet.ch3;

printf("\nOriginal Number: %u\n", originalNumber);

return 0;

}

**OUPUT:**



**Q5.** 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.

**CODE:**

#include <stdio.h>

int isLittleEndian() {

unsigned int num = 1;

return \*((unsigned char\*)&num);

}

unsigned int convertEndianness(unsigned int num) {

return ((num >> 24) & 0xFF) | ((num >> 8) & 0xFF00) | ((num << 8) & 0xFF0000) | ((num << 24) & 0xFF000000);

}

int main() {

unsigned int number;

printf("Enter a number (unsigned integer): ");

scanf("%u", &number);

printf("Memory Address -> value");

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

unsigned char \*ptr = (unsigned char \*)&number;

for (int i = 0; i < sizeof(number); i++) {

printf("\n%d: \t\t%02X", &ptr[i], ptr[i]);

}

if (isLittleEndian()) {

printf("\n\nThe host machine is in Little Endian.\n");

number = convertEndianness(number);

} else {

printf("\n\nThe host machine is in Big Endian.\n");

number = convertEndianness(number);

}

printf("\nConverted Number: %u\n", number);

return 0;

}

**OUTPUT:**

