**IOT Assignment 1**

Name: Vinay Saunshi Date: 26/08/2025

USN: 1RVU23CSE539

Q1. Given the following variables:

int i = 100;

float f = 200.5;

char c = 'A';

Explain what happens when you execute: c = \*(char \*)(&f); What value will 'c' hold and why?

Ans: **What happens:**

* &f gets the address of the float variable f.
* (char \*)(&f) casts that address to a char \* (pointer to a single byte).
* \*(char \*)(&f) accesses the first byte of the float variable f.
* This first byte (which is part of the internal binary representation of 200.5) is assigned to c.

**Value of c:**

* c holds the first byte of the binary representation of 200.5 in memory.
* This is not the ASCII character '200.5', but some binary byte value.
* The exact character depends on the machine's endianness (byte order).
* Usually, c will hold some non-printable or strange character because it’s raw binary data.

Q2. Write a C program where you store a string in a character array and then use a pointer to change the third character using pointer arithmetic.

Ans: Changing the third character of a string using pointer arithmetic.

#include <stdio.h>  
int main() {  
 char str[] = "HELLO";  
 char \*p = str;  
 \*(p + 2) = 'X'  
 printf("%s\n", str);   
 return 0;  
}

**Output:** HEXLO

Q3. Consider the following code:

int x = 300;

char \*cp = (char \*)&x;

\*cp = 65;

Predict the value of 'x' after execution and explain your reasoning.

Ans: **Code**

int x = 300;  
char \*cp = (char \*)&x;  
\*cp = 65;

**Explanation:**

* x = 300 stored in 4 bytes (example: in little endian: 0x2C 0x01 0x00 0x00)
* cp points to the first byte of x
* \*cp = 65 sets the first byte to 65 (ASCII 'A', decimal 65 = 0x41)
* Now, the new bytes of x are (0x41 0x01 0x00 0x00)
* In decimal, new value of x = 0x0141 (little endian) = 1\*256 + 65 = 321

**Value of x after execution:**

321

Q4. Given a float variable 'fp' and a char pointer 'cp', demonstrate how you can make 'cp' point to the second byte of 'fp' using pointer arithmetic. Explain how the byte ordering of your machine affects the result.

Ans: Point cp to the second byte of a float variable fp.

**Example:**

float fp = 123.45;  
char \*cp = (char \*)&fp;  
cp = cp + 1;

**Explanation:**

* cp initially points to the first byte of fp.
* Adding 1 moves cp to point to the **second byte**.
* Byte ordering (endianness) affects which part of the float cp now points to:
  + **Little endian:** cp points to the second least significant byte.
  + **Big endian:** cp points to the second most significant byte.
* This changes how you interpret the value if you try to read or manipulate it.

Q5. Write a program that: - Declares an int, float, and char variable - Uses a void pointer to point to each of them - Prints their values using pointer casting. Discuss the importance of correct casting when using void pointers.

Ans: Using void pointer with int, float, char

#include <stdio.h>  
  
int main() {  
 int i = 10;  
 float f = 20.5;  
 char c = 'Z';  
 void \*ptr;  
  
 ptr = &i;  
 printf("int: %d\n", \*(int \*)ptr);  
  
 ptr = &f;  
 printf("float: %.2f\n", \*(float \*)ptr);  
  
 ptr = &c;  
 printf("char: %c\n", \*(char \*)ptr);  
 return 0;  
}

**Output:**

int: 10  
float: 20.50  
char: Z

**Importance of casting:**

* void \* is a generic pointer with no type information.
* You **must cast** it back to the correct pointer type to correctly interpret the data.
* Incorrect casting leads to wrong data reading and bugs.

Q6. Given:

char str[] = "HELLO";

char \*p = str; \*(p+1) = 'A';

What will be the output of printf("%s", str)? Explain why.

Ans: **Code**

char str[] = "HELLO";  
char \*p = str;  
\*(p + 1) = 'A';  
printf("%s", str);

**Explanation:**

* p points to str.
* \*(p + 1) = 'A'; changes the second character (index 1) from 'E' to 'A'.
* So, str becomes "HALLO".

**Output:**

HALLO

**IOT Assignment 2**

**IOT Assignment 3**

