Experiment 2 Lab 1: Understanding Union vs Structure

main.c

Share Run

```c
#include <stdio.h>
#include <string.h>

// Define a structure to store employee information
struct EmployeeStruct {
    char name[50];
    int employeeID;
    float salary;
};

// Define a union to store employee information
union EmployeeUnion {
    char name[50];
    int employeeID;
    float salary;
};

int main() {
    // Declare and initialize structure
    struct EmployeeStruct empStruct;
    strcpy(empStruct.name, "Alice");
    empStruct.employeeID = 1234;
    empStruct.salary = 55000.50;

    // Declare and initialize union
    union EmployeeUnion empUnion;
    strcpy(empUnion.name, "Alice");
    empUnion.employeeID = 1234;  // Overwrites name
    empUnion.salary = 55000.50;   // Overwrites employeeID

    // Display values stored in structure
    printf("Using structure:\n");
    printf("Name: %s\n", empStruct.name);
    printf("Employee ID: %d\n", empStruct.employeeID);
    printf("Salary: %.2f\n", empStruct.salary);

    // Display values stored in union
    printf("\nUsing union:\n");
    printf("Name: %s\n", empUnion.name);    // Name will not be displayed correctly due to overwriting
    printf("Employee ID: %d\n", empUnion.employeeID);  // This value will also not be correct
    printf("Salary: %.2f\n", empUnion.salary);  // Only the last field is valid

    // Display memory size occupied by structure and union
    printf("\nSize of structure: %lu bytes\n", sizeof(empStruct));
    printf("Size of union: %lu bytes\n", sizeof(empUnion));

    return 0;
}
```

Output

```
/tmp/IAoUooVGfI.o
Using structure:
Name: Alice
Employee ID: 1234
Salary: 55000.50

Using union:
Name: ��VGe
Employee ID: 1196873856
Salary: 55000.50

Size of structure: 60 bytes
Size of union: 52 bytes


=== Code Execution Successful ===
```

Experiment 2 Lab 2: Dynamic Memory Allocation using malloc() and free()

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>  // For malloc() and free()

int main() {
    int n;
    int *arr;
    int sum = 0;
    float average;

    // Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Dynamically allocate memory using malloc()
    arr = (int *)malloc(n * sizeof(int));

    // Check if memory allocation was successful
    if (arr == NULL) {
        printf("Memory allocation failed!\n");
        return 1;  // Exit if memory allocation failed
    }

    // Input elements into the array
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Calculate the sum of the elements
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    // Calculate the average
    average = (float)sum / n;

    // Output the sum and average
    printf("Sum: %d\n", sum);
    printf("Average: %.2f\n", average);

    // Release the dynamically allocated memory
    free(arr);

    return 0;
}
```

Output

```
/tmp/3d6inQmvGR.o
Enter the number of elements: 4
Enter 4 elements:
1
2
3
4
Sum: 10
Average: 2.50


=== Code Execution Successful ===
```