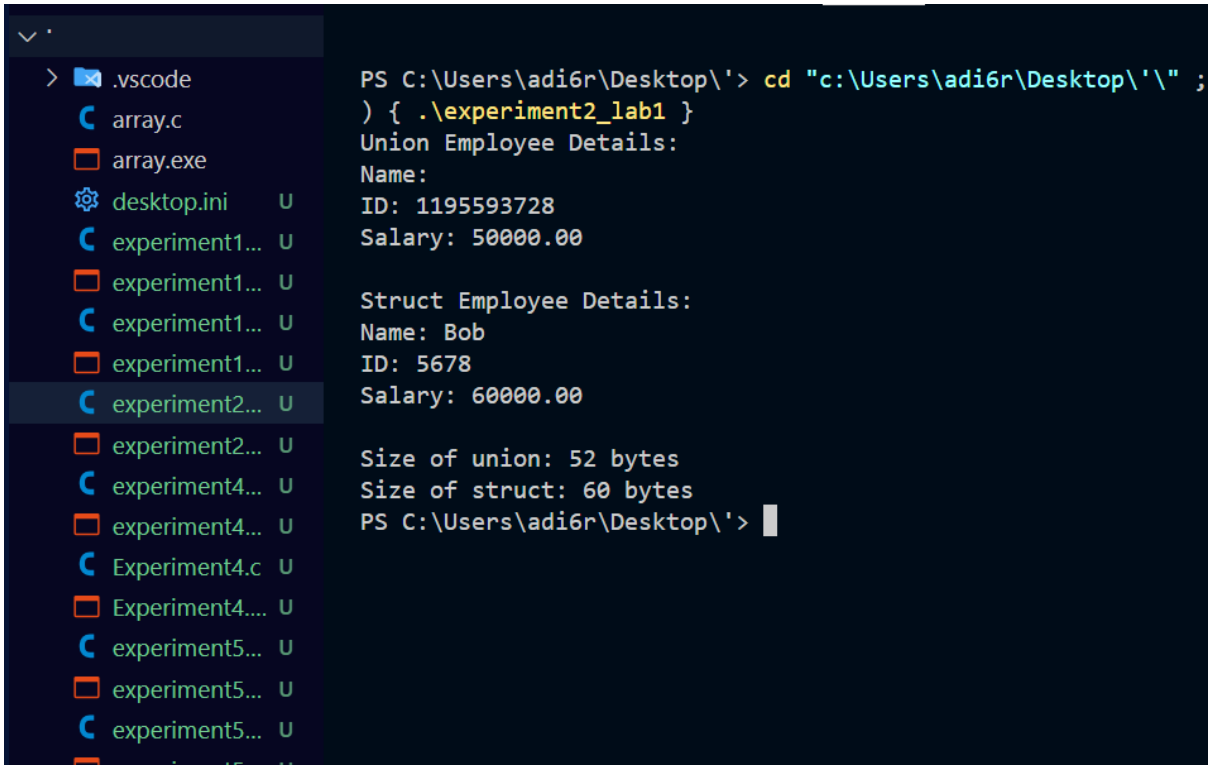
Student Information

- Name:- Aditya Kumar
- Sap Id :- 590015145
- Branch :- M.C.A
- Batch :- B1
- Instructor :-Dr. Sourbh Kumar

Lab Assignment 1: Understanding Union vs Structure

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

union Employee {
    char name[50];
    int id;
    float salary;
};

struct EmployeeInfo {
    char name[50];
    int id;
    float salary;
};

int main() {
    union Employee emp1;
    struct EmployeeInfo emp2;

    // Assign values to union members
    strcpy(emp1.name, "Alice");
    emp1.id = 1234;
    emp1.salary = 50000.0;

    // Assign values to struct members
    strcpy(emp2.name, "Bob");
    emp2.id = 5678;
    emp2.salary = 60000.0;

    printf("Union Employee Details:\n");
    printf("Name: %s\n", emp1.name);
    printf("ID: %d\n", emp1.id);
    printf("Salary: %.2f\n", emp1.salary);
```

```
    printf("\nStruct Employee Details:\n");
    printf("Name: %s\n", emp2.name);
    printf("ID: %d\n", emp2.id);
    printf("Salary: %.2f\n", emp2.salary);

    printf("\nSize of union: %zu bytes\n", sizeof(emp1));
    printf("Size of struct: %zu bytes\n", sizeof(emp2));

    return 0;
}
```



Lab Assignment 2: Dynamic Memory Allocation with malloc() and free()

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n;

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

```
    // Dynamically allocate memory for an array of n integers
    int *arr = (int*)malloc(n * sizeof(int));

    if (arr == NULL)
     {
        printf("Memory allocation failed!\n");
        return 1;
      }

    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int sum = 0;
    for (int i = 0; i < n; i++)
   {
        sum += arr[i];
    }
    float average = (float)sum / n;
    printf("Sum of elements: %d\n", sum);
    printf("Average of elements: %.2f\n", average);

    // Free the allocated memory
    free(arr);

    return 0;
}
```