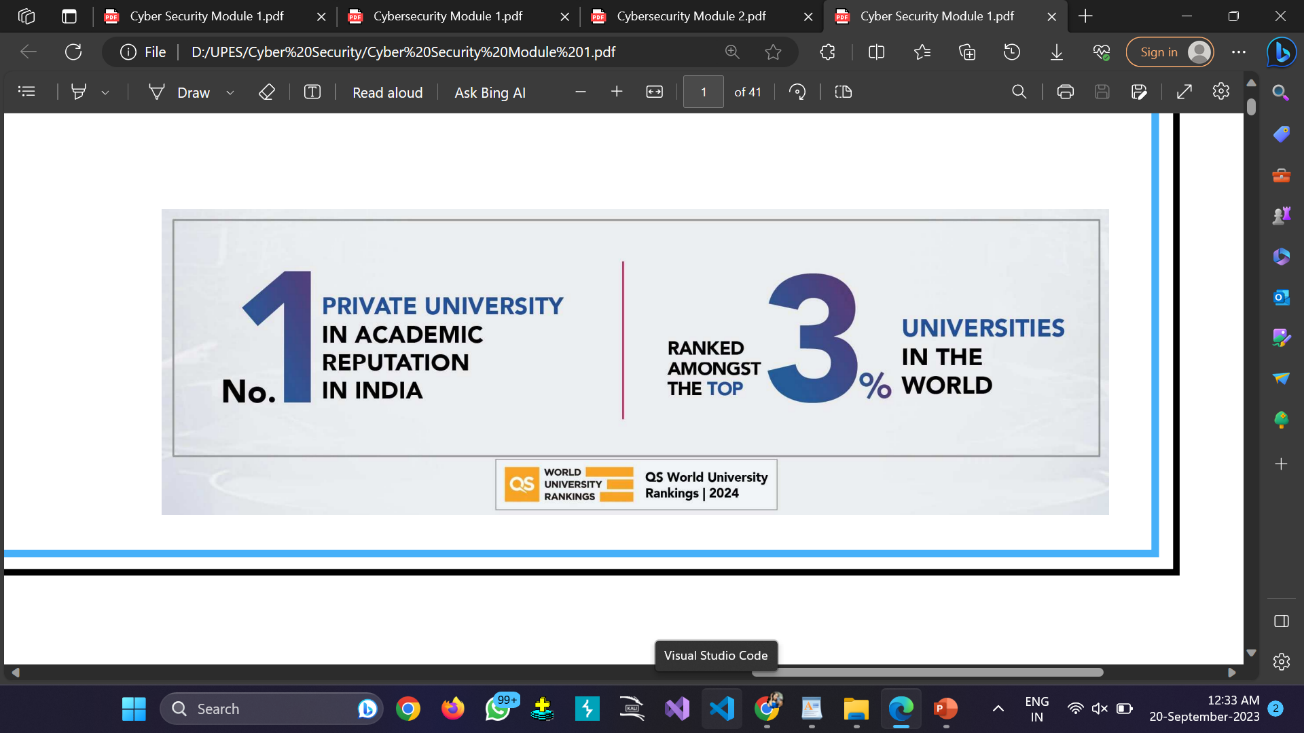
A picture containing text, clipart

Description automatically generated

1

**Lab Experiment: 02**

**Student Detail:**

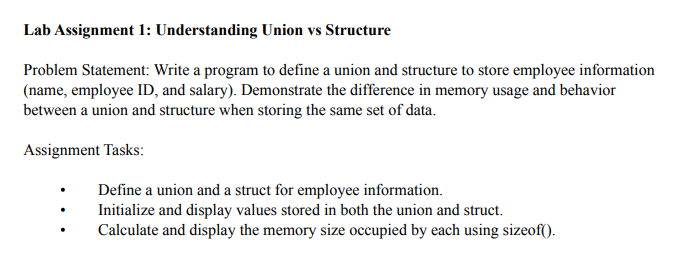
**• Name:** Prashant Joshi

**• Student ID:** 590010879

**• Branch:** MCA

**• Batch:** B1

**• Instructor:** Dr. Sourbh Kumar



Solution:

#include <stdio.h>

#include <string.h>

struct EmployeeStruct {

char name[50]; // Name of the employee

int employeeID; // Employee ID

float salary; // Employee salary

};

union EmployeeUnion {

char name[50]; // Name of the employee

int employeeID; // Employee ID

float salary; // Employee salary

};

int main() {

// Initialize structure

struct EmployeeStruct empStruct;

strcpy(empStruct.name, "Rahul Kumar"); // employee name

empStruct.employeeID = 12345;

empStruct.salary = 50000.50;

// Initialize union

union EmployeeUnion empUnion;

strcpy(empUnion.name, "Rahul Kumar"); // employee name

empUnion.employeeID = 12345;

empUnion.salary = 50000.50;

// Display structure values

printf("Structure - Employee Information:\n");

printf("Name: %s\n", empStruct.name);

printf("Employee ID: %d\n", empStruct.employeeID);

printf("Salary: %.2f\n", empStruct.salary);

// Display union values

printf("\nUnion - Employee Information:\n");

printf("Name: %s\n", empUnion.name); // Note: Only the last written field will be valid

printf("Employee ID: %d\n", empUnion.employeeID);

printf("Salary: %.2f\n", empUnion.salary);

// Display memory size of structure and union

printf("\nMemory Size:\n");

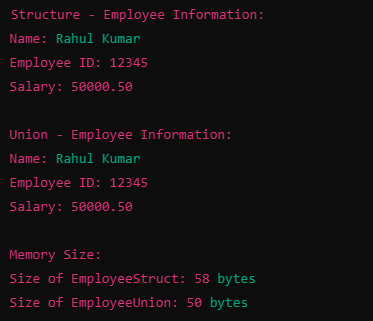
printf("Size of EmployeeStruct: %lu bytes\n", sizeof(empStruct));

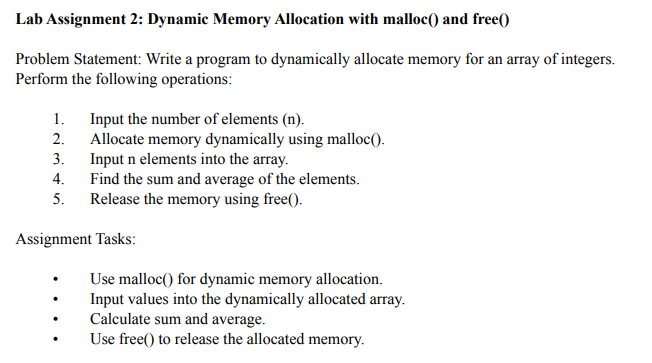
printf("Size of EmployeeUnion: %lu bytes\n", sizeof(empUnion));

return 0;

}

Output:





Solution:

#include <stdio.h>

#include <stdlib.h> // for malloc() and free()

int main() {

int n, i;

int \*arr;

int sum = 0;

float average;

// Step 1: Input the number of elements

printf("Enter the number of elements: ");

scanf("%d", &n);

// Step 2: Dynamically allocate memory using malloc()

arr = (int \*)malloc(n \* sizeof(int)); // Allocate memory for n integers

// Check if memory allocation was successful

if (arr == NULL) {

printf("Memory allocation failed!\n");

return 1; // Exit the program if memory allocation fails

}

// Step 3: Input n elements into the array

printf("Enter %d integers:\n", n);

for (i = 0; i < n; i++) {

printf("Element %d: ", i + 1);

scanf("%d", &arr[i]);

}

// Step 4: Calculate the sum and average of the elements

for (i = 0; i < n; i++) {

sum += arr[i];

}

average = (float)sum / n;

// Display the results

printf("\nSum of elements: %d\n", sum);

printf("Average of elements: %.2f\n", average);

// Step 5: Free the dynamically allocated memory

free(arr);

printf("\nMemory successfully freed.\n");

return 0;

}

Output:

