**Batch: C4-1 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 7**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:**  Write a program in C to demonstrate use of structures and union. |

**AIM:** Write a program to manage an employee database using structure and union in C. Each Employee has the following information:

1. Employee ID(integer)
2. Name(string)
3. Department(string)
4. Salary(float)

You need to implement the following functionalities:

1. Create a structure named Employee with the appropriate data members to store the information mentioned above.
2. Create a union named EmployeeInfo that can hold either the Name or Department information.
3. Write a function addEmployee that takes user input for each employee's information and stores it in an array of structures.
4. Write a function printEmployeeDetails that takes an employee's ID as input and prints all available details for that employee.
5. Write a function updateEmployeeInfo that takes an employee's ID and allows the user to update either the Name or Department information using the EmployeeInfo union.
6. Implement a menu-driven program that allows the user to perform the above operations. Include options to add a new employee, print employee details, update employee information, and exit the program.

\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

Design modular programs using functions and the use of structure and union(CO4).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

The program accepts a choice from the user using a switch case statement and generates output accordingly.

**Algorithm:**

Structures and Unions:

1. De­fine a structure called Employe­e with these fie­lds:

- Employee ID (integer)

- Name­ (string)

- Department (string)

- Salary (float)

2. Define­ a union called EmployeeInfo with the­se fields:

- Name (string)

- Department (string)

- Salary (float)

Functions:

3. The main() function:

- Declare­ an array of Employee structures calle­d employees with a maximum size of 100­.

- Initialize variables like the­ number of employee­s, user's choice, and employe­e ID.

- Display a main menu using a do-while loop:

- Show me­nu options:

- 1. Add Employee

- 2. Print Employee­ Details

- 3. Update Employee­ Information

- 4. Exit

- Get the user's input for the­ir choice.

- Use a switch-case for e­ach menu option:

- Case 1: Call the addEmploye­e() function.

- Case 2: Call the printEmploye­eDetails() function.

- Case 3: Call the­ updateEmployeeInfo() function.

- Case­ 4: Exit the program.

- Default: Show an "Invalid choice" me­ssage.

- Repeat the loop until the user chooses to exit (choice 4).

4. Function addEmployee(struct Employee employees[], int \*numEmployees):

- Check if numEmployees is less than 100.

- If true, get input for new employee's:

- empID

- name

- department

- salary

- Increment numEmployees.

- Display "Employee added successfully".

- If false, display "Maximum number of employees reached".

5. Function printEmployeeDetails(struct Employee employees[], int numEmployees, int empID):

- Iterate through employees array:

- If empID matches an employee's empID, print:

- Employee ID

- Name

- Department

- Salary

- Return.

- If no match found, display "Employee with ID not found".

6. Function updateEmployeeInfo(struct Employee employees[], int numEmployees, int empID):

- Declare variables choice, info, employeeFound as integers.

- Iterate through employees array:

- If empID matches an employee's empID, set employeeFound to 1.

- Display menu for updating:

- 1. Update Name

- 2. Update Department

- 3. Update Salary

- Get choice from user.

- Use switch-case on choice:

- Case 1: Get new name and update employee's name.

- Case 2: Get new department and update employee's department.

- Case 3: Get new salary and update employee's salary.

- Default: Display "Invalid choice".

- Break loop once employee is updated.

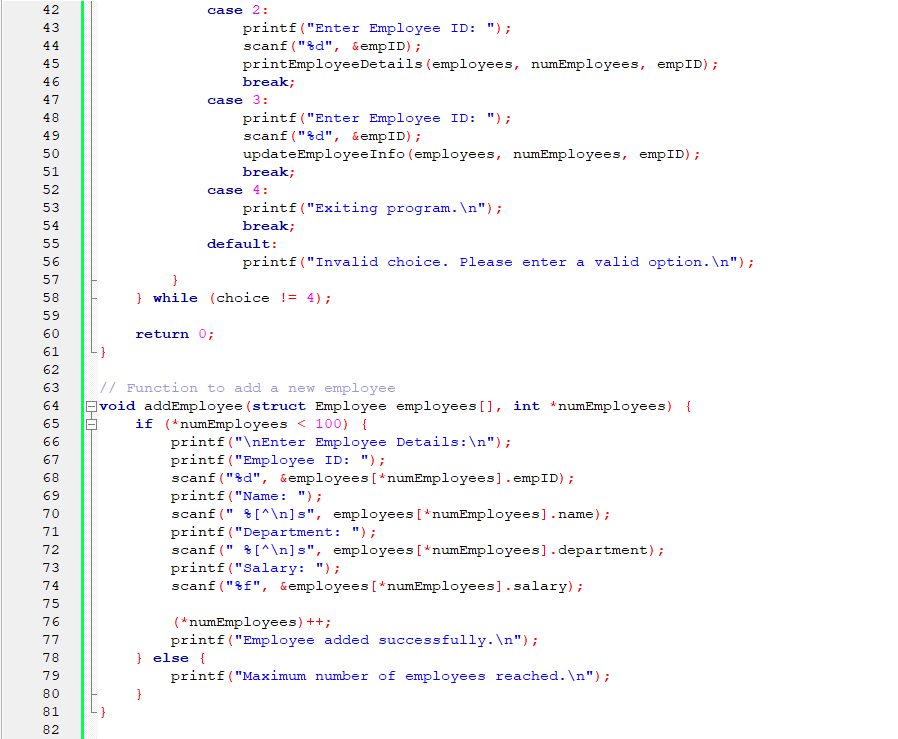
- If employeeFound is 0,

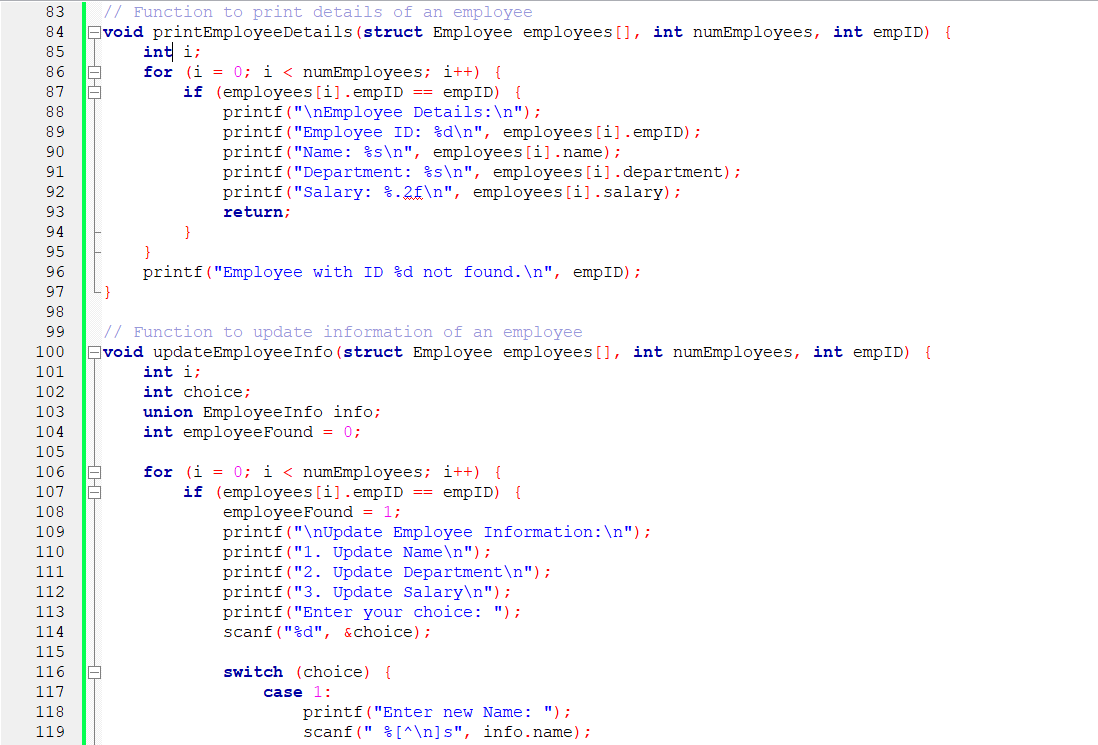
display "Employee with ID not found".

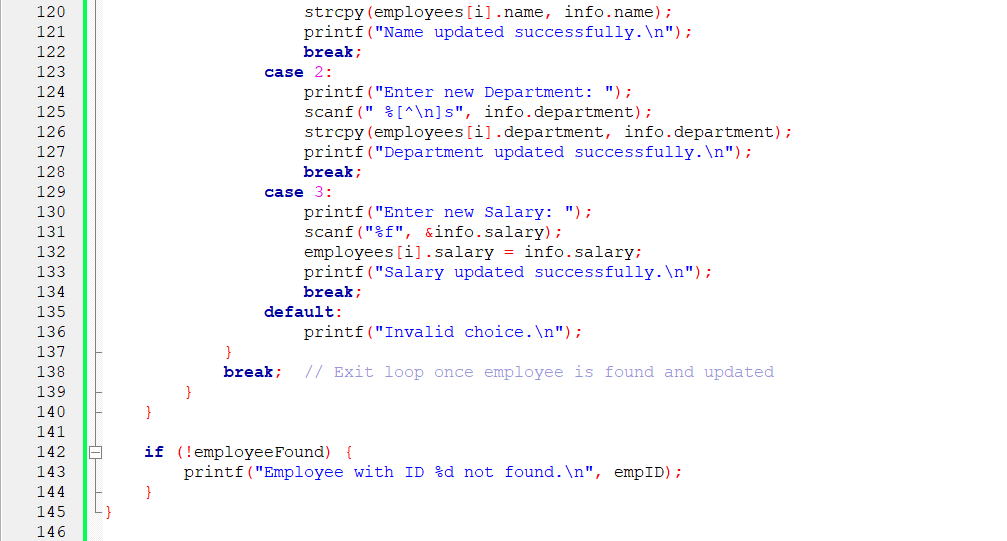
**Implementation details:**

Code:

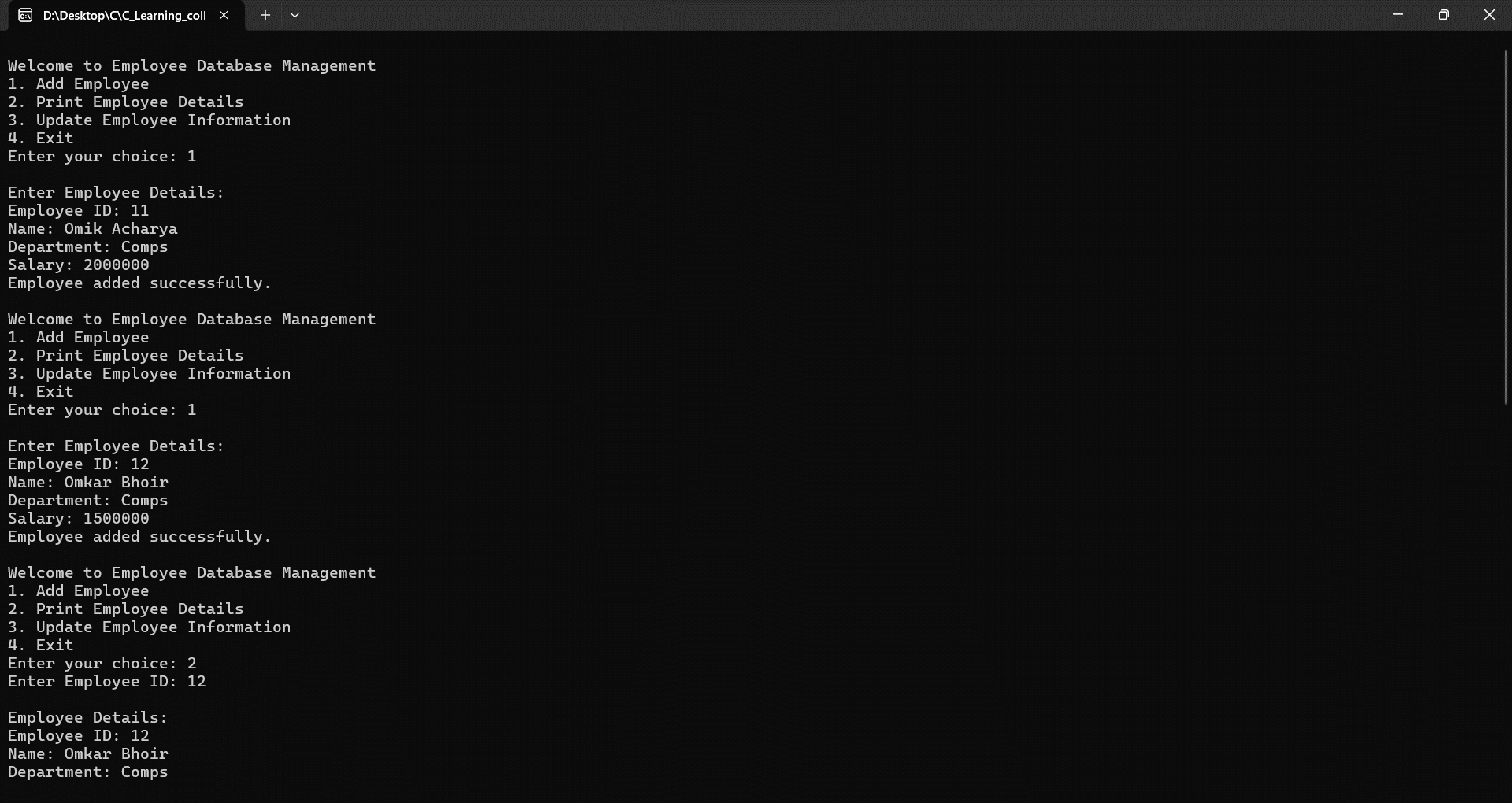








**Output(s):**





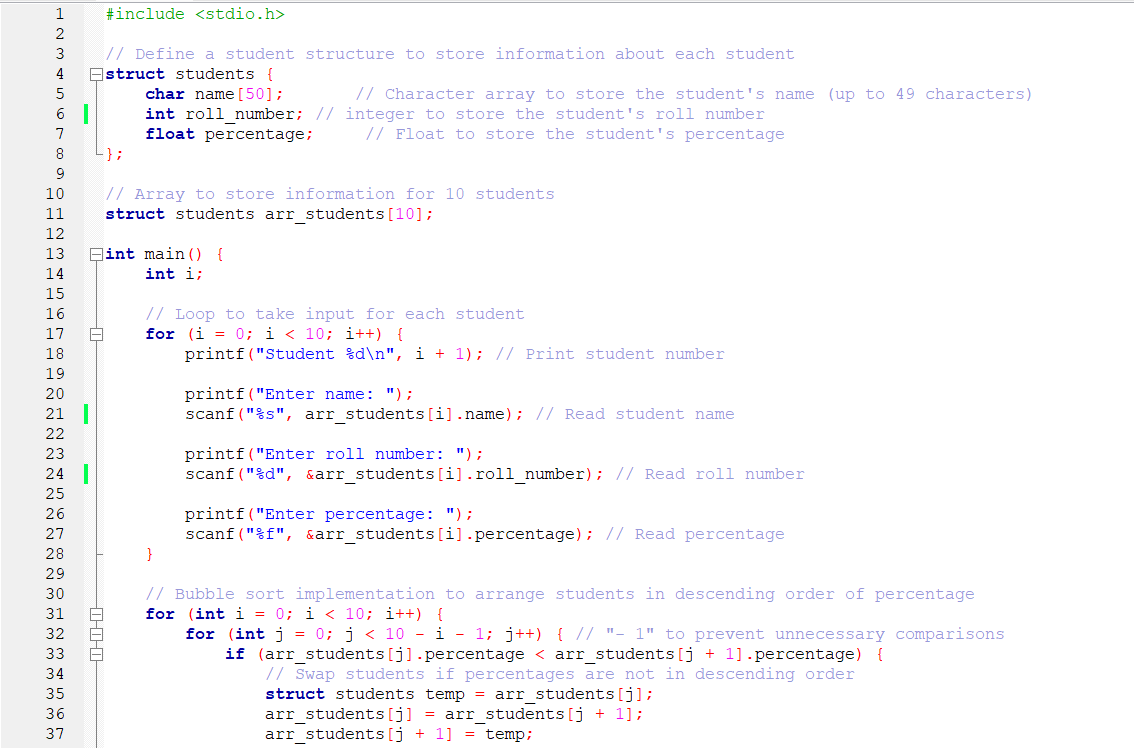
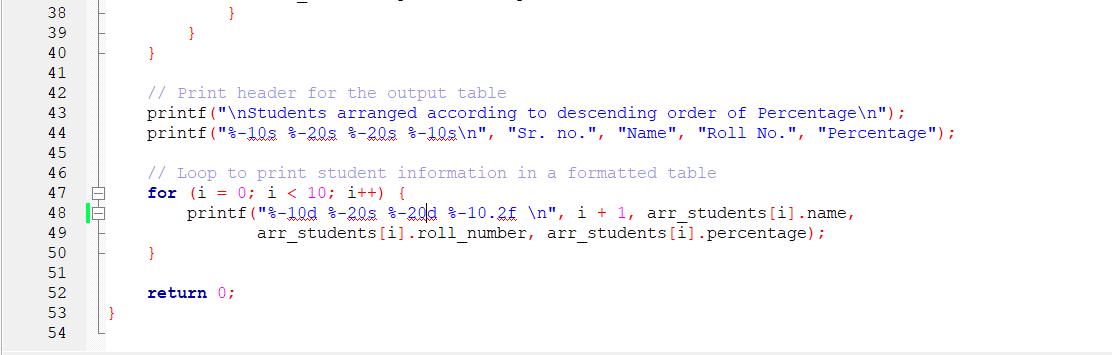
**Conclusion:**

In this C programming project, we built an Employee Database Management program using structures and unions. The program lets you add new employees, view their details, and update their information like names, departments, and salaries. Through this project, we learned how to use structures to organize employee data efficiently and unions for flexible data storage. The program's menu-driven interface makes it easy for users to interact with, reinforcing our understanding of modular design and user input handling. In the future, we could improve the program by adding input validation, deletion functionality, and file handling for data persistence, to enhance its functionality and usability. This project demonstrates the practical application of structures and unions in C programming for effectively managing and manipulating employee data.

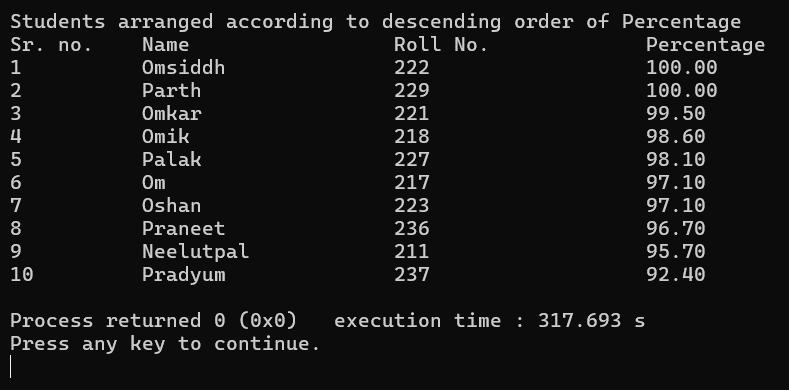
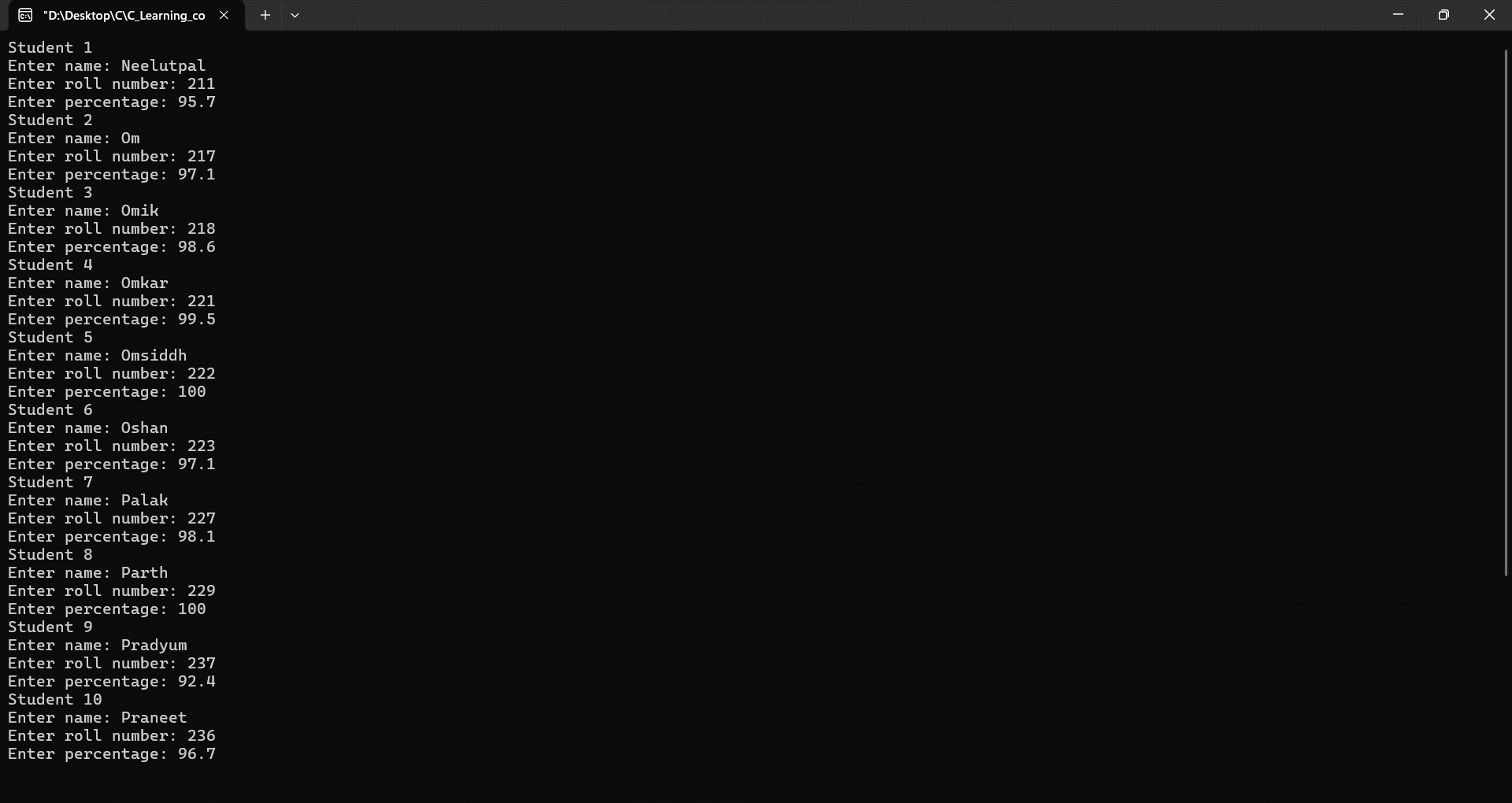
**Post Lab Descriptive Questions**

* WAP to accept student name, roll number and percentage for 10 students using array of structures and arrange them in descending order of their percentage.

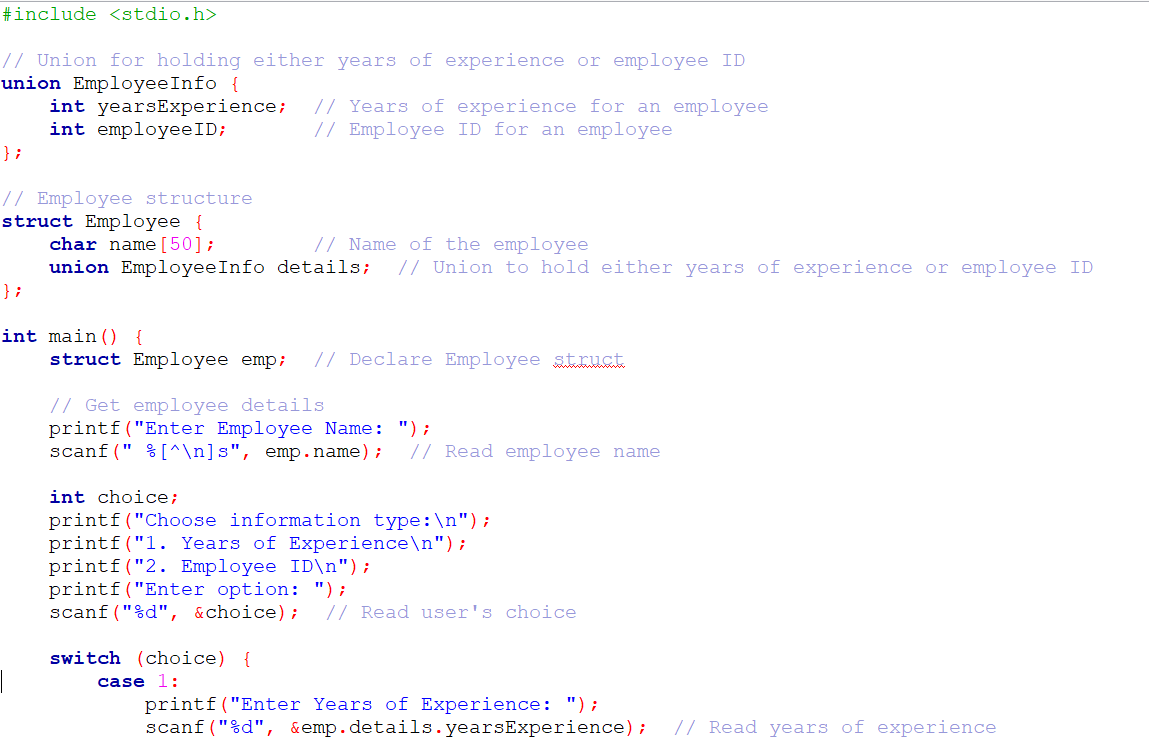
Code:

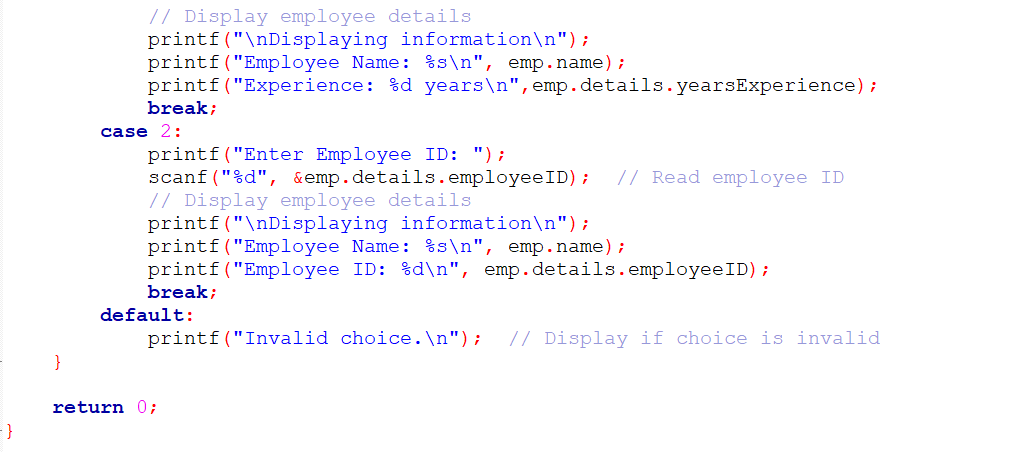
 

Output:

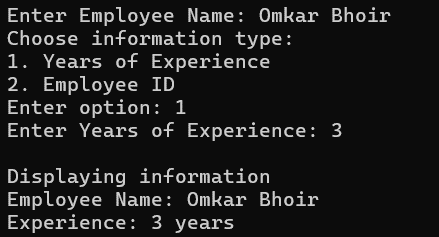


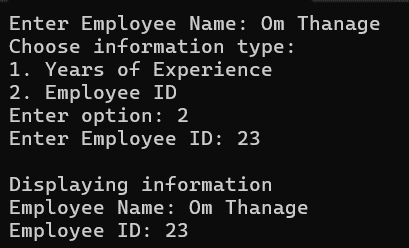
* WAP to display employee name, ID and year of experience using union.

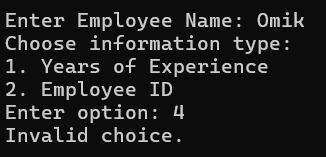




Output:







**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**