Batch: C1-2 Roll No.: 16010123036

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:

```
1. Start

2. Declare a structure 'Employee' with integer 'id', character arrays 'name' and 'dept',
and float 'salary'.

3. Declare an array 'database' of 'Employee' structures with size 100.

4. Declare an integer variable 'n' and initialize it to 0 to keep track of the number of
employees.

5. Define a function 'addEmployee' to add employee details to the database:

   a. Check if the database is full (n == 99):

      i. If true, print "DATABASE FULL!!" and return.

   b. Prompt the user to enter employee details: ID, Name, Department, and Salary.

   c. Store the entered details in the next available index of the 'database' array.

   d. Increment 'n' to indicate the addition of an employee.

6. Define a function 'printEmployeeDetails' to print details of a specific employee:

   a. If 'id' is -1, prompt the user to enter the employee ID.

   b. Iterate through the 'database' array to find the employee with the given ID:
```

i. If found, print the employee details (ID, Name, Department, and Salary) and return the index of the employee.

c. If not found, print "NOT FOUND!!" and return -1.

7. Define a function 'updateEmployeeInfo' to update employee details:

a. Call 'printEmployeeDetails' to get the index of the employee to update.

b. If the employee is not found, return.

c. Display a menu to choose which detail to edit: Name, Department, Salary, or Exit.

d. Based on the user's choice, prompt for the new value and update the corresponding detail of the employee.

8. Define the 'main' function:

a. Display a welcome message.

b. Enter a loop to repeatedly prompt the user for actions:

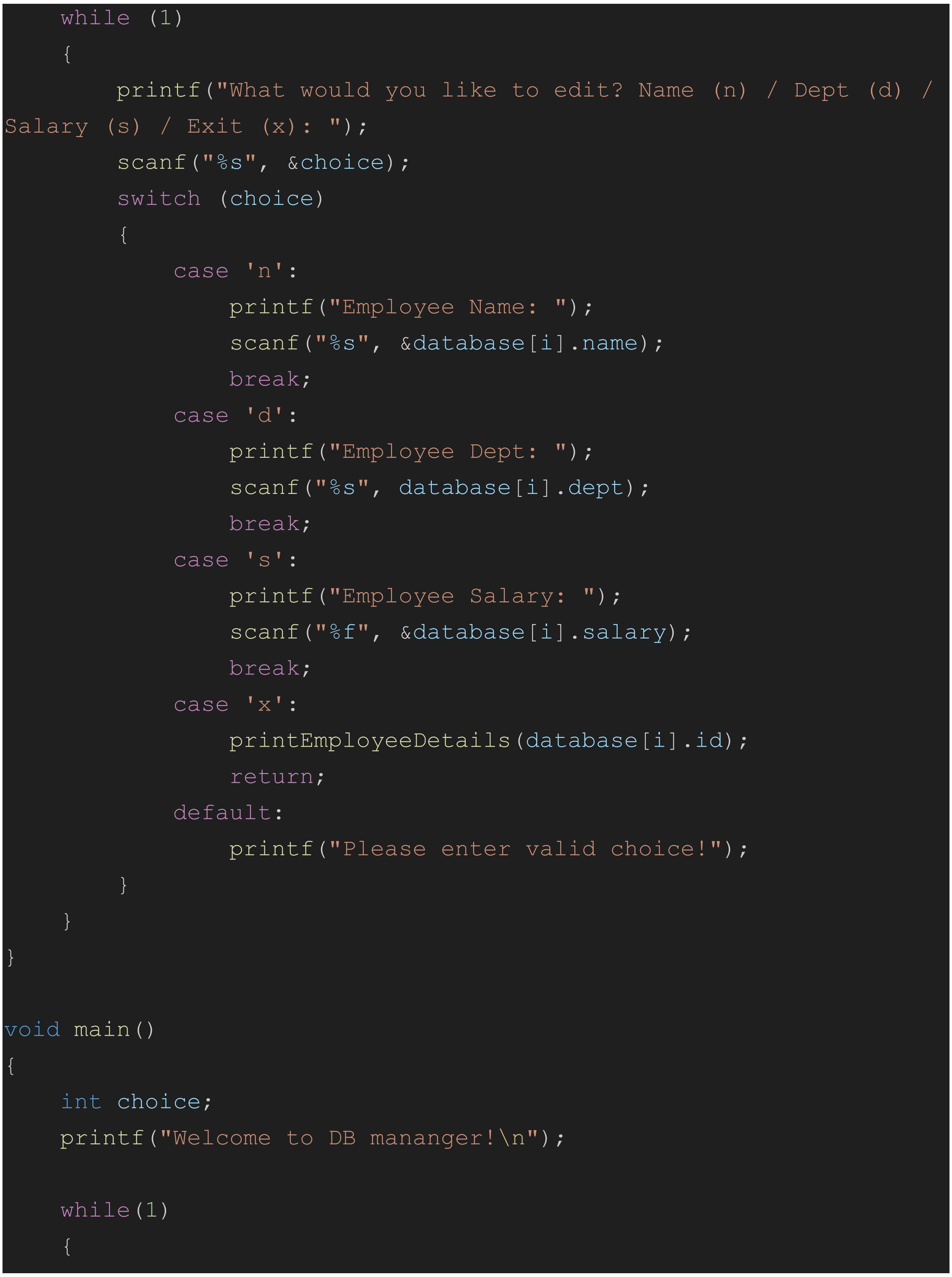
i. Prompt the user to choose an action: Add Employee, Print Employee Details, Update Employee Details, or Exit.

ii. Based on the choice, call the corresponding function: 'addEmployee', 'printEmployeeDetails', 'updateEmployeeInfo', or exit the program.

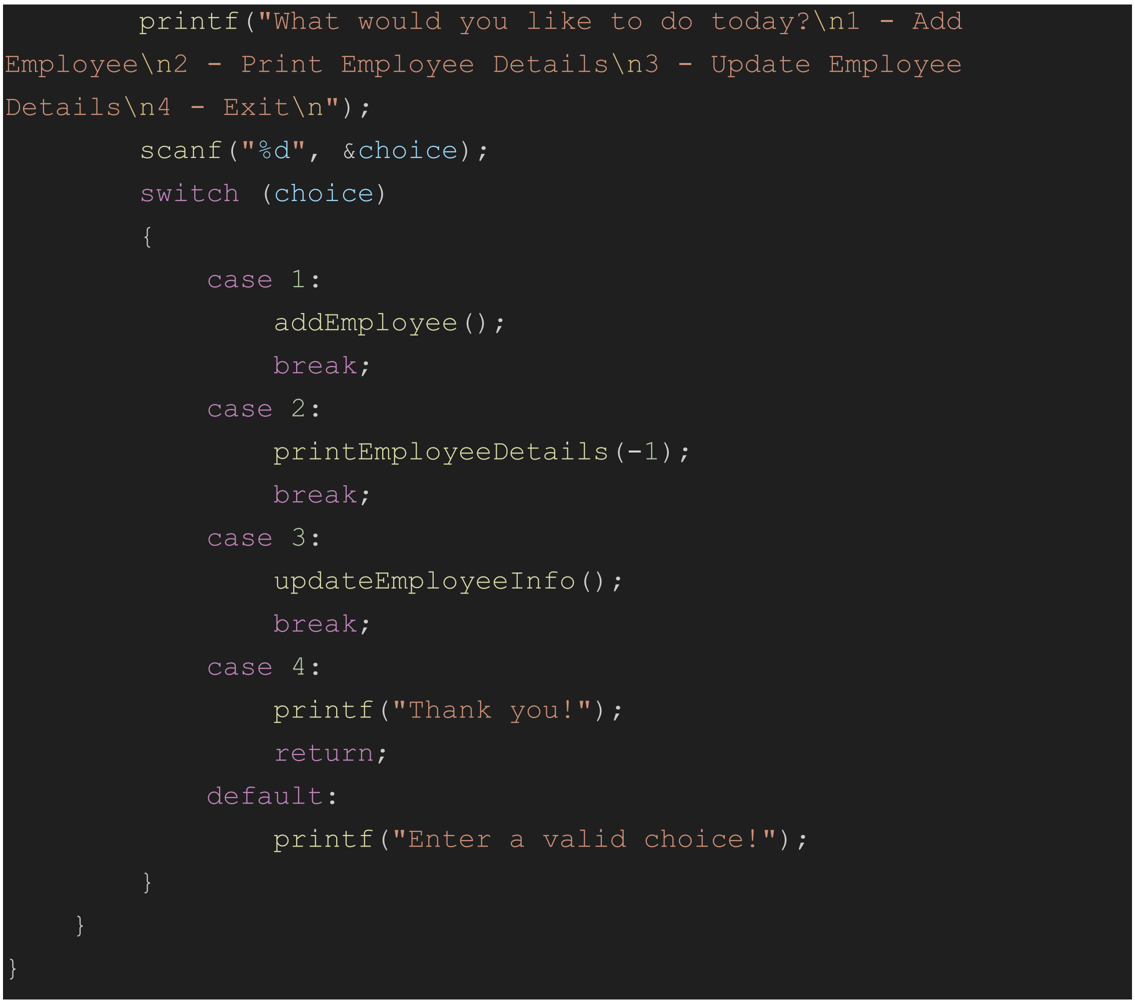
9. End

Implementation details:

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

struct Employee {
    int id;
    char name[50];
    char dept[50];
    float salary;
} database[100];

int n = 0;

void addEmployee()
{
    if (n == 99)
    {
        printf("DATABASE FULL!!");
        return;
    }

    int id;
    char name[50];
    char dept[50];
    float salary;

    printf("Employee ID: ");
    scanf("%d", &id);
    printf("Employee Name: ");
    scanf("%s", name);
    printf("Employee Dept: ");
    scanf("%s", dept);
    printf("Employee Salary: ");
    scanf("%f", &salary);

    database[n].id = id;
    strcpy(database[n].name, name);
    strcpy(database[n].dept, dept);
    database[n].salary = salary;
    n++;
    return;
}

int printEmployeeDetails(int id)
{
    if (id == -1)
    {
        printf("Enter ID: ");
        scanf("%d", &id);
    }

    for (int i = 0; i <= n; i++)
    {
        if (database[i].id == id)
        {
            printf("Employee ID: %d\n", database[i].id);
            printf("Employee Name: %s\n", database[i].name);
            printf("Employee Dept: %s\n", database[i].dept);
            printf("Employee Salary: %f\n", database[i].salary);
            return i;
        }
    }
    printf("NOT FOUND!!");
    return -1;
}

void updateEmployeeInfo()
{
    int i = printEmployeeDetails(-1);

    if (i == -1) return;

    char choice;

```



```
    while (1)
    {
        printf("What would you like to edit? Name (n) / Dept (d) /
Salary (s) / Exit (x): ");
        scanf("%s", &choice);
        switch (choice)
        {
            case 'n':
                printf("Employee Name: ");
                scanf("%s", &database[i].name);
                break;
            case 'd':
                printf("Employee Dept: ");
                scanf("%s", database[i].dept);
                break;
            case 's':
                printf("Employee Salary: ");
                scanf("%f", &database[i].salary);
                break;
            case 'x':
                printEmployeeDetails(database[i].id);
                return;
            default:
                printf("Please enter valid choice!");
        }
    }
}

void main()
{
    int choice;
    printf("Welcome to DB mananger!\n");

    while(1)
    {
```



```
        printf("What would you like to do today?\n1 - Add 
Employee\n2 - Print Employee Details\n3 - Update Employee 
Details\n4 - Exit\n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                addEmployee();
                break;
            case 2:
                printEmployeeDetails(-1);
                break;
            case 3:
                updateEmployeeInfo();
                break;
            case 4:
                printf("Thank you!");
                return;
            default:
                printf("Enter a valid choice!");
        }
    }
}
```

Output(s):

```
Welcome to DB mananger!
What would you like to do today?
1 - Add Employee
2 - Print Employee Details
3 - Update Employee Details
4 - Exit
1
Employee ID: 1234
Employee Name: Amandeep
Employee Dept: Comps
Employee Salary: 150000
What would you like to do today?
```

```
1 - Add Employee
2 - Print Employee Details
3 - Update Employee Details
4 - Exit
2
Enter ID: 1234
Employee ID: 1234
Employee Name: Amandeep
Employee Dept: Comps
Employee Salary: 150000.000000
What would you like to do today?
1 - Add Employee
2 - Print Employee Details
3 - Update Employee Details
4 - Exit
3
Enter ID: 1234
Employee ID: 1234
Employee Name: Amandeep
Employee Dept: Comps
Employee Salary: 150000.000000
What would you like to edit? Name (n) / Dept (d) / Salary (s) / Exit (x): s
Employee Salary: 200000
What would you like to edit? Name (n) / Dept (d) / Salary (s) / Exit (x): x
Employee ID: 1234
Employee Name: Amandeep
Employee Dept: Comps
Employee Salary: 200000.000000
What would you like to do today?
1 - Add Employee
2 - Print Employee Details
3 - Update Employee Details
4 - Exit
2
Enter ID: 1234
Employee ID: 1234
Employee Name: Amandeep
Employee Dept: Comps
Employee Salary: 200000.000000
What would you like to do today?
1 - Add Employee
2 - Print Employee Details
3 - Update Employee Details
```

4 - Exit
4
Thank you!

Conclusion:
In this experiment, we utilized C structures and unions to manage an employee database efficiently. Structures stored employee details like ID, name, department, and salary, while unions offered flexibility for name or department storage.

The program included functions for adding new employees, printing details by ID, and updating information. These features were organized for clarity.

The menu-driven interface facilitated user interaction, enhancing usability. This experiment demonstrates the practical utility of structures and unions in programming tasks.

Post Lab Descriptive Questions

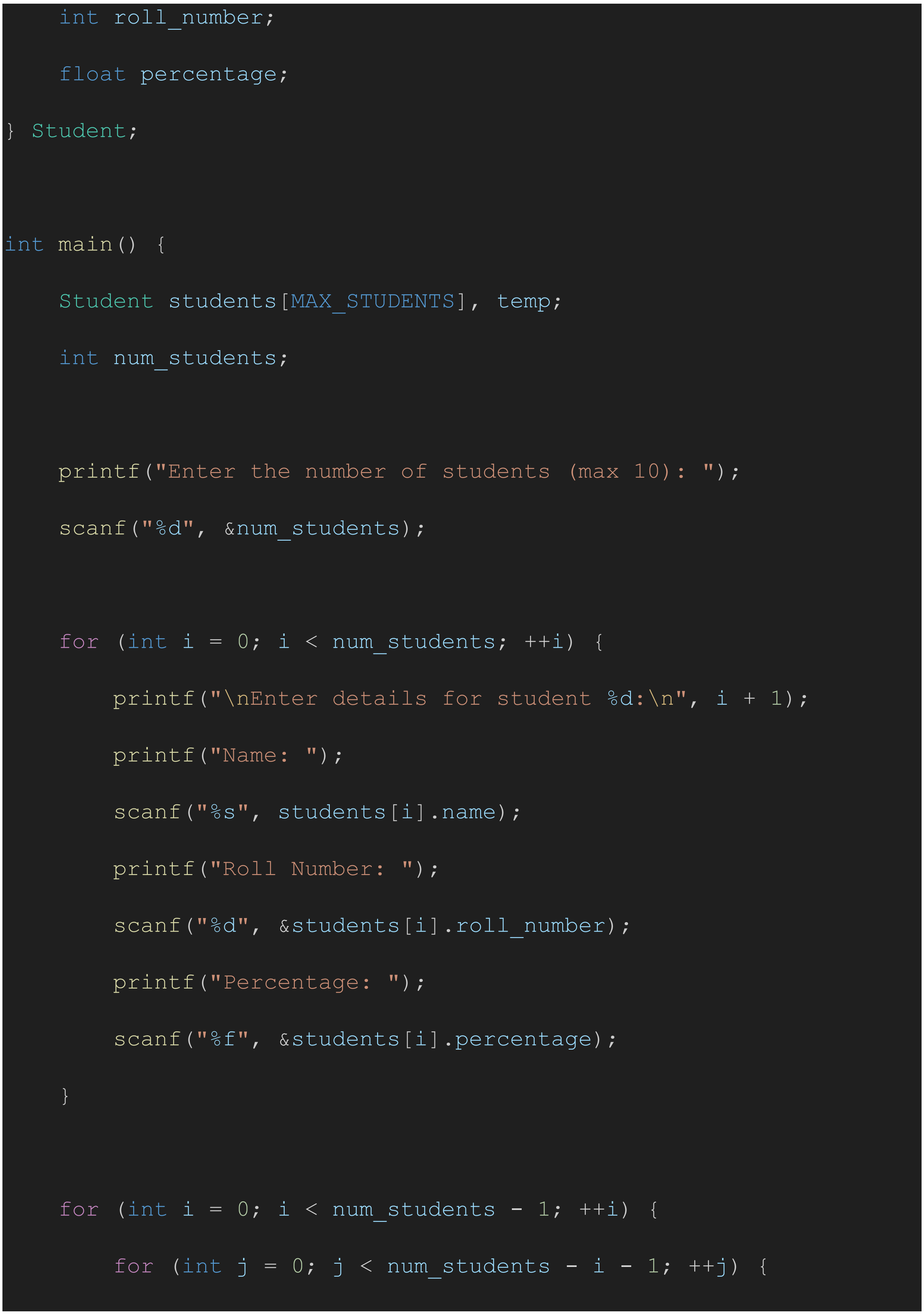
- WAP to accept student name, roll number and percentage for 10 students using array of structures and arrange them in descending order of their percentage.
- WAP to display employee name, ID and year of experience using union.
- Virtual lab on Structure and Union

  https://cse02-iiith.vlabs.ac.in/exp/structures/simulation.html

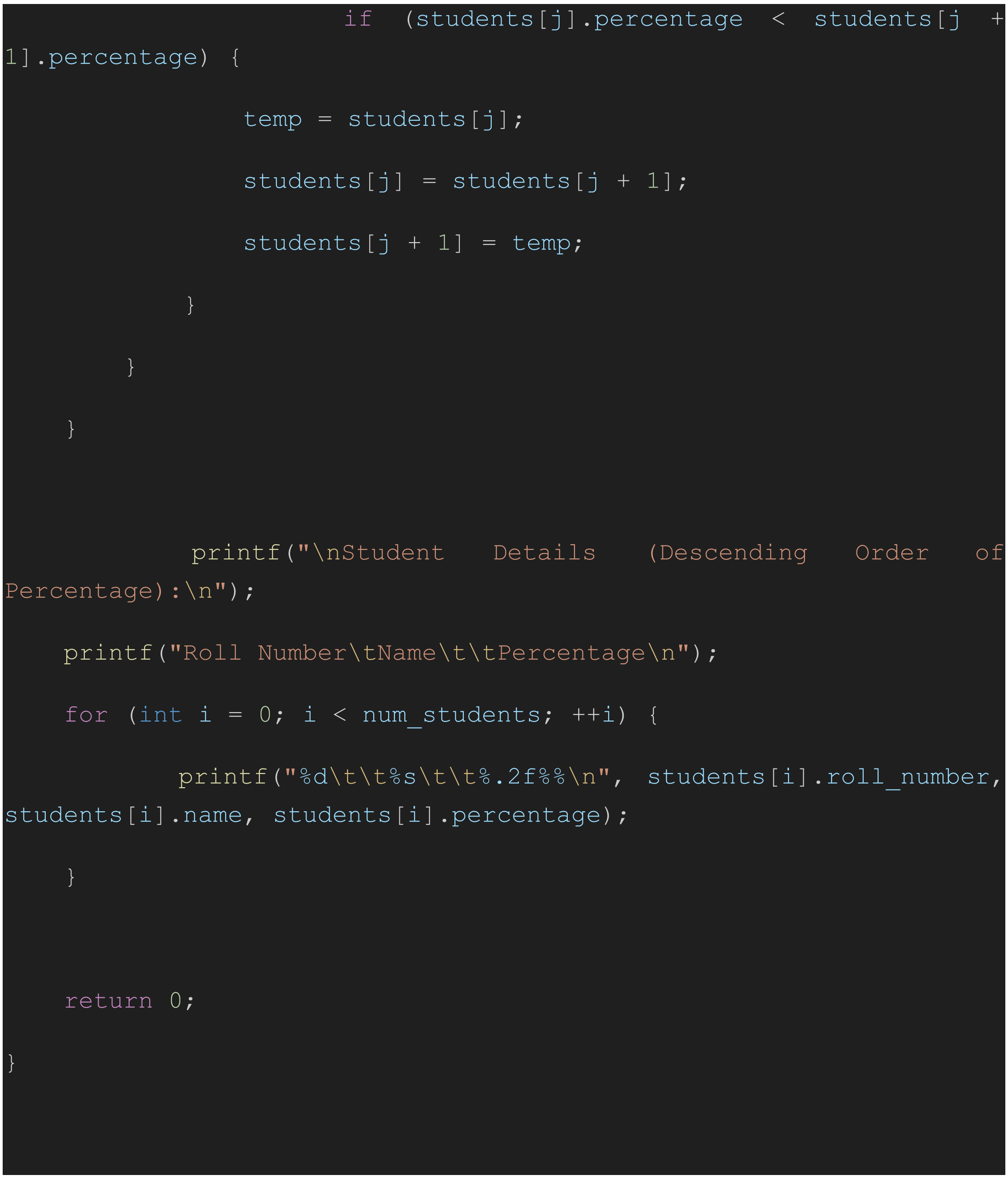
Answers:

1)

```
#include <stdio.h>

#include <string.h>



#define MAX_STUDENTS 10

#define MAX_NAME_LENGTH 50



typedef struct {

    char name[MAX_NAME_LENGTH];
```



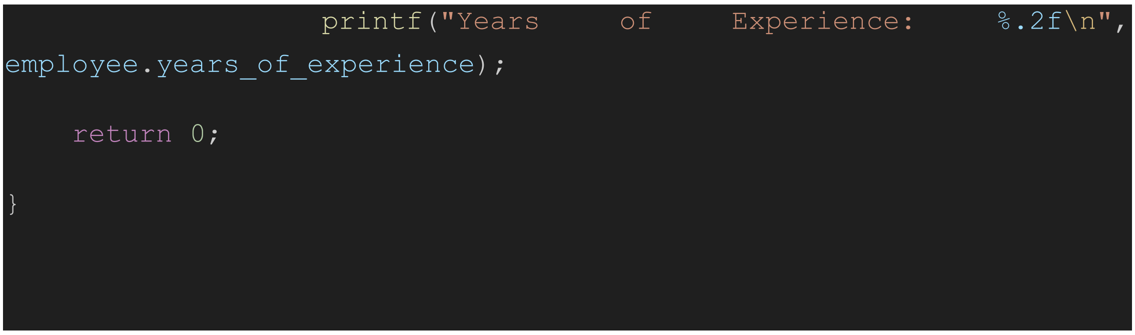
```
    int roll_number;
    float percentage;
} Student;

int main() {
    Student students[MAX_STUDENTS], temp;
    int num_students;

    printf("Enter the number of students (max 10): ");
    scanf("%d", &num_students);

    for (int i = 0; i < num_students; ++i) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", students[i].name);
        printf("Roll Number: ");
        scanf("%d", &students[i].roll_number);
        printf("Percentage: ");
        scanf("%f", &students[i].percentage);
    }

    for (int i = 0; i < num_students - 1; ++i) {
        for (int j = 0; j < num_students - i - 1; ++j) {
```



```
            if (students[j].percentage < students[j + 1].percentage) {
                temp = students[j];
                students[j] = students[j + 1];
                students[j + 1] = temp;
            }
        }
    }

    printf("\nStudent Details (Descending Order of Percentage):\n");
    printf("Roll Number\tName\t\tPercentage\n");
    for (int i = 0; i < num_students; ++i) {
        printf("%d\t\t%s\t\t%.2f%%\n", students[i].roll_number, students[i].name, students[i].percentage);
    }

    return 0;
}
```

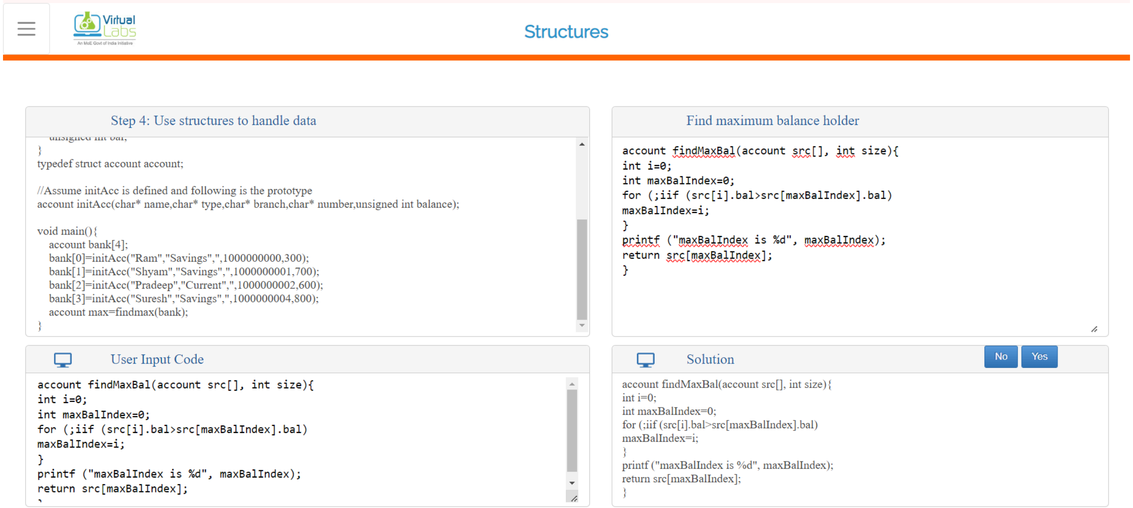
2)

```
#include <stdio.h>

union EmployeeInfo {

    char name[50];

    int id;

    float years_of_experience;

};



int main() {

    union EmployeeInfo employee;



    printf("Enter employee name: ");

    scanf("%s", employee.name);



    printf("Enter employee ID: ");

    scanf("%d", &employee.id);



    printf("Enter years of experience: ");

    scanf("%f", &employee.years_of_experience);



    printf("\nEmployee Details:\n");

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

    printf("ID: %d\n", employee.id);
```



```
                printf("Years    of    Experience:    %.2f\n",
employee.years_of_experience);

    return 0;

}
```

3)



Step 4: Use structures to handle data

```
    unsigned int bal;
}
typedef struct account account;

//Assume initAcc is defined and following is the prototype
account initAcc(char* name,char* type,char* branch,char* number,unsigned int balance);

void main(){
   account bank[4];
   bank[0]=initAcc("Ram","Savings",",1000000000,300);
   bank[1]=initAcc("Shyam","Savings",",1000000001,700);
   bank[2]=initAcc("Pradeep","Current",",1000000002,600);
   bank[3]=initAcc("Suresh","Savings",",1000000004,800);
   account max=findmax(bank);
}
```

Find maximum balance holder

```
account findMaxBal(account src[], int size){
int i=0;
int maxBalIndex=0;
for (;iif (src[i].bal>src[maxBalIndex].bal)
maxBalIndex=i;
}
printf ("maxBalIndex is %d", maxBalIndex);
return src[maxBalIndex];
}
```

User Input Code

```
account findMaxBal(account src[], int size){
int i=0;
int maxBalIndex=0;
for (;iif (src[i].bal>src[maxBalIndex].bal)
maxBalIndex=i;
}
printf ("maxBalIndex is %d", maxBalIndex);
return src[maxBalIndex];
}
```

Solution

```
account findMaxBal(account src[], int size){
int i=0;
int maxBalIndex=0;
for (;iif (src[i].bal>src[maxBalIndex].bal)
maxBalIndex=i;
}
printf ("maxBalIndex is %d", maxBalIndex);
return src[maxBalIndex];
}
```

Date: ____________ **Signature of faculty in-charge**