UCS 2201 Fundamentals and Practice of Software Development

A7: Programs using Structures, Pointers and File

Batch 2023-2027
Name: SK Shivaanee
Register ID: 2310257
Section: CSE B
Dr. Chitra Babu, Dr. D. Thenmozhi, Dr. M. Saritha

Learning Outcome: You will be able to implement user defined datatypes in C with the following features: ● Using structures ● Passing structures to a function ● Using pointers ● Using Files

You will be able to adapt to the following best practices ● Modular programming and incremental programming ● Using user defined data types ● Multi-file program with user defined header files

Assignment: Write the algorithm and solve the following problems by implementing in C. (CO7, K3, 1.3.1, 1.4.1, 2.1.2, 2.1.3, 2.4.3, 3.2.2, 3.4.3, 4.1.2, 4.2.1, 5.2.2, 13.2.1, 13.3.2, 13.4.2, 14.2.1, 14.2.2)

1. Define a datatype for Employee with members Emp_Id, Emp_Name, DOB, Age, Address, Dept, Basic_Salary, Allowance[3], Deduction[2], Gross_Salary and Net_Salary. Define DOB with members namely Day, Month and Year. Define Address with members namely Door_No, Street, Area, City and Pincode. Let the Allowance array consists of Dearness_Allowance, HRA and Medical_Allowance. The Deduction array consists of PF and Income_Tax. Write a C program using structures to perform the following operations.
   a. Write a function to create an employee database using an array of structures with 5 employees belonging to 2 departments by passing the structure name and number of employees as arguments to the functions. Get the input from users only for Emp_Id, Emp_Name, DOB, Address, Dept and Basic_Salary.
   b. Write a function to find the Age of an employee using DOB and the current Date by passing DOB as argument.
   c. Write a function for calculating Allowances using the following
   i. Dearness_Allowance = 42% of Basic_Salary
   ii. HRA = 10% of Basic_Salary
   iii. Medical_Allowance = 15% of Basic_Salary
   d. Write a function for calculating the Gross_Salary as Basic_Salary + Allowances
   e. Write a function for calculating Deductions using the following
   i. PF = 12% of Basic_Salary
   ii. Income_Tax = 20% of Gross_Salary
   f. Write a function for calculating the Net_Salary as Basic_Salary + Allowances - Deductions
   g. Write a function to search for an employee based on the Emp_Id and display his/her payslip.
   h. Write a function to display the department that pays the highest salary to an employee

Structures used:

```
→ typedef struct
  {
      int Day;
      int Month;
      int Year;
  } DOB;

→ typedef struct
  {
      char Door_No[10];
      char Street[50];
      char Area[50];
      char City[50];
      int Pincode;
  } Address;

→ typedef struct
  {
      int Emp_Id;
      char Emp_Name[50];
      DOB dob;
      int Age;
      Address address;
      char Dept[50];
      float Basic_Salary;
      float Allowance[3];
      float Deduction[2];
      float Gross_Salary;
      float Net_Salary;
  } Employee
```

Functions used:

```
→ void calculateAge (Employee* emp, DOB current_date);
→ void calculateAllowances (Employee *emp);
→ void calculateGrossSalary (Employee *emp);
→ void calculateDeductions (Employee* emp);
→ void calculateNetSalary (Employee * emp);
→ void createEmployeeDatabase (Employee emp[], int num)
→ void searchEmployeeById (Employee emp[], int num, int id)
→ void highestPayingDepartment (Employee emp[], int num)
```

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

typedef struct {
    int Day;
    int Month;
    int Year;
} DOB;

typedef struct {
    char Door_No[10];
    char Street[50];
    char Area[50];
    char City[50];
    int Pincode;
} Address;

typedef struct {
    int Emp_Id;
    char Emp_Name[50];
    DOB dob;
    int Age;
    Address address;
    char Dept[50];
    float Basic_Salary;
    float Allowance[3];
    float Deduction[2];
    float Gross_Salary;
    float Net_Salary;
} Employee;

void calculateAge(Employee* emp, DOB current_date) {
    int age = current_date.Year - emp->dob.Year;
    if (current_date.Month < emp->dob.Month ||
       (current_date.Month == emp->dob.Month && current_date.Day < emp->dob.Day)) {
        age--;
    }
    emp->Age = age;
}

void calculateAllowances(Employee* emp) {
    emp->Allowance[0] = emp->Basic_Salary * 0.42;
    emp->Allowance[1] = emp->Basic_Salary * 0.10;
```

```c
    emp->Allowance[1] = emp->Basic_Salary * 0.10;
    emp->Allowance[2] = emp->Basic_Salary * 0.15;
}

void calculateGrossSalary(Employee* emp) {
    emp->Gross_Salary = emp->Basic_Salary + emp->Allowance[0] + emp->Allowance[1] + emp->Allowance[2];
}

void calculateDeductions(Employee* emp) {
    emp->Deduction[0] = emp->Basic_Salary * 0.12;
    emp->Deduction[1] = emp->Gross_Salary * 0.20;
}

void calculateNetSalary(Employee* emp) {
    emp->Net_Salary = emp->Gross_Salary - emp->Deduction[0] - emp->Deduction[1];
}

void createEmployeeDatabase(Employee emp[], int num) {
    for (int i = 0; i < num; i++) {
        printf("Enter details for employee %d\n", i + 1);
        printf("Emp_Id: ");
        scanf("%d", &emp[i].Emp_Id);
        printf("Emp_Name: ");
        scanf(" %[^\n]", emp[i].Emp_Name);
        printf("DOB (Day Month Year): ");
        scanf("%d %d %d", &emp[i].dob.Day, &emp[i].dob.Month, &emp[i].dob.Year);
        printf("Address (Door_No Street Area City Pincode): ");
        scanf("%s %s %s %s %d", emp[i].address.Door_No, emp[i].address.Street, emp[i].address.Area, emp[i].address.City, &emp[i].address.Pincode);
        printf("Dept: ");
        scanf("%s", emp[i].Dept);
        printf("Basic_Salary: ");
        scanf("%f", &emp[i].Basic_Salary);

        // Calculate Age
        DOB current_date = {2, 6, 2024}; // Set the current date here
        calculateAge(&emp[i], current_date);

        // Calculate Allowances, Gross Salary, Deductions, and Net Salary
        calculateAllowances(&emp[i]);
        calculateGrossSalary(&emp[i]);
        calculateDeductions(&emp[i]);
        calculateNetSalary(&emp[i]);
```

```c
        calculateNetSalary(&emp[i]);
    }
}

void searchEmployeeById(Employee emp[], int num, int id) {
    for (int i = 0; i < num; i++) {
        if (emp[i].Emp_Id == id) {
            printf("Employee found:\n");
            printf("Emp_Id: %d\n", emp[i].Emp_Id);
            printf("Emp_Name: %s\n", emp[i].Emp_Name);
            printf("DOB: %d-%d-%d\n", emp[i].dob.Day, emp[i].dob.Month, emp[i].dob.Year);
            printf("Age: %d\n", emp[i].Age);
            printf("Address: %s, %s, %s, %s, %d\n", emp[i].address.Door_No, emp[i].address.Street, emp[i].address.Area, emp[i].address.City, emp[i].address.Pincode);
            printf("Dept: %s\n", emp[i].Dept);
            printf("Basic_Salary: %.2f\n", emp[i].Basic_Salary);
            printf("Dearness_Allowance: %.2f\n", emp[i].Allowance[0]);
            printf("HRA: %.2f\n", emp[i].Allowance[1]);
            printf("Medical_Allowance: %.2f\n", emp[i].Allowance[2]);
            printf("Gross_Salary: %.2f\n", emp[i].Gross_Salary);
            printf("PF: %.2f\n", emp[i].Deduction[0]);
            printf("Income_Tax: %.2f\n", emp[i].Deduction[1]);
            printf("Net_Salary: %.2f\n", emp[i].Net_Salary);
            return;
        }
    }
    printf("Employee with ID %d not found.\n", id);
}

void highestPayingDepartment(Employee emp[], int num) {
    char highest_dept[50];
    float max_salary = 0;
    for (int i = 0; i < num; i++) {
        if (emp[i].Net_Salary > max_salary) {
            max_salary = emp[i].Net_Salary;
            strcpy(highest_dept, emp[i].Dept);
        }
    }
    printf("The department that pays the highest salary is: %s\n", highest_dept);
}

int main() {
    int num_employees = 5;
```

```c
    int num_employees = 5;
    Employee employees[num_employees];

    createEmployeeDatabase(employees, num_employees);

    int search_id;
    printf("Enter the Emp_Id to search: ");
    scanf("%d", &search_id);
    searchEmployeeById(employees, num_employees, search_id);

    highestPayingDepartment(employees, num_employees);

    return 0;
}
```

```
Enter details for employee 1
Emp_Id: 1234
Emp_Name: shiv
DOB (Day Month Year): 11 12 1999
Address (Door_No Street Area City Pincode): 3 7 wer sdf 600119
Dept: it
Basic_Salary: 150000
Enter details for employee 2
Emp_Id: 2345
Emp_Name: ram
DOB (Day Month Year): 04 07 1989
Address (Door_No Street Area City Pincode): 4 7 qwe sdf 600115
Dept: chem
Basic_Salary: 100000
Enter details for employee 3
Emp_Id: 1456
Emp_Name: ravi
DOB (Day Month Year): 04 06 1989
Address (Door_No Street Area City Pincode): 6 5 dfg nnv 600032
Dept: comm
Basic_Salary: 75000
Enter details for employee 4
Emp_Id: 6789
Emp_Name: suresh
DOB (Day Month Year): 17 09 1976
Address (Door_No Street Area City Pincode): 9 4 dfg asd 600078
Dept: tech
Basic_Salary: 95000
Enter details for employee 5
Emp_Id: 2398
Emp_Name: rahul
DOB (Day Month Year): 16 04 1987
Address (Door_No Street Area City Pincode): 23 7 yui xcv 600045
Dept: comm
Basic_Salary: 90000
Enter the Emp_Id to search: 6789
Employee found:
Emp_Id: 6789
Emp_Name: suresh
DOB: 17-9-1976
Age: 47
Address: 9, 4, dfg, asd, 600078
Dept: tech
Basic_Salary: 95000.00
Dearness_Allowance: 39900.00
HRA: 9500.00
Medical_Allowance: 14250.00
Gross_Salary: 158650.00
PF: 11400.00
Income_Tax: 31730.00
Net_Salary: 115520.00
The department that pays the highest salary is: it
```

2. Modify Qn 1 by using files to perform the following a. Read the details namely Emp_Id, Emp_Name, DOB, Address, Dept and Basic_Salary of 5 employees and store them in a file. b. Access the file sequentially to get the employee details for computing Gross and Net salaries as mentioned in Qn 1. c. Create 5 files consisting of the payslips of 5 employees

Structures used:

→ Same as Question 1.

Functions used:

→ void calculateAge (Employee * emp, DOB current_date);

→ void calculateAllowances (Employee * emp);

→ void calculateGrossSalary (Employee* emp);

→ void calculateDeductions (Employee * emp);

→ void calculateNetSalary (Employee * emp);

→ void createEmployeeDatabaseFile(const char * filename);

→ void computeSalaries (const char * filename);

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

typedef struct {
    int Day;
    int Month;
    int Year;
} DOB;

typedef struct {
    char Door_No[10];
    char Street[50];
    char Area[50];
    char City[50];
    int Pincode;
} Address;

typedef struct {
    int Emp_Id;
    char Emp_Name[50];
    DOB dob;
    int Age;
    Address address;
    char Dept[50];
    float Basic_Salary;
    float Allowance[3];
    float Deduction[2];
    float Gross_Salary;
    float Net_Salary;
} Employee;

void calculateAge(Employee* emp, DOB current_date) {
    int age = current_date.Year - emp->dob.Year;
    if (current_date.Month < emp->dob.Month ||
       (current_date.Month == emp->dob.Month && current_date.Day < emp->dob.Day)) {
        age--;
    }
    emp->Age = age;
}

void calculateAllowances(Employee* emp) {
    emp->Allowance[0] = emp->Basic_Salary * 0.42;
    emp->Allowance[1] = emp->Basic_Salary * 0.10;
```

```c
    emp->Allowance[2] = emp->Basic_Salary * 0.15;
}

void calculateGrossSalary(Employee* emp) {
    emp->Gross_Salary = emp->Basic_Salary + emp->Allowance[0] + emp->Allowance[1] + emp->Allowance[2];
}

void calculateDeductions(Employee* emp) {
    emp->Deduction[0] = emp->Basic_Salary * 0.12;
    emp->Deduction[1] = emp->Gross_Salary * 0.20;
}

void calculateNetSalary(Employee* emp) {
    emp->Net_Salary = emp->Gross_Salary - emp->Deduction[0] - emp->Deduction[1];
}

void createEmployeeDatabaseFile(const char* filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        perror("Error opening file");
        return;
    }

    Employee emp;
    for (int i = 0; i < 5; i++) {
        printf("Enter details for employee %d\n", i + 1);
        printf("Emp_Id: ");
        scanf("%d", &emp.Emp_Id);
        printf("Emp_Name: ");
        scanf(" %[^\n]", emp.Emp_Name);
        printf("DOB (Day Month Year): ");
        scanf("%d %d %d", &emp.dob.Day, &emp.dob.Month, &emp.dob.Year);
        printf("Address (Door_No Street Area City Pincode): ");
        scanf("%s %s %s %s %d", emp.address.Door_No, emp.address.Street, emp.address.Area, emp.address.City, &emp.address.Pincode);
        printf("Dept: ");
        scanf("%s", emp.Dept);
        printf("Basic_Salary: ");
        scanf("%f", &emp.Basic_Salary);

        fwrite(&emp, sizeof(Employee), 1, file);
    }
```

```c
    fclose(file);
}

void computeSalaries(const char* filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file");
        return;
    }

    FILE *payslip_file;
    char payslip_filename[50];
    Employee emp;
    DOB current_date = {2, 6, 2024}; // Set the current date here

    while (fread(&emp, sizeof(Employee), 1, file)) {
        // Calculate Age
        calculateAge(&emp, current_date);

        // Calculate Allowances, Gross Salary, Deductions, and Net Salary
        calculateAllowances(&emp);
        calculateGrossSalary(&emp);
        calculateDeductions(&emp);
        calculateNetSalary(&emp);

        // Create payslip file
        sprintf(payslip_filename, "payslip_%d.txt", emp.Emp_Id);
        payslip_file = fopen(payslip_filename, "w");
        if (payslip_file == NULL) {
            perror("Error opening payslip file");
            continue;
        }

        fprintf(payslip_file, "Employee Payslip\n");
        fprintf(payslip_file, "---------------\n");
        fprintf(payslip_file, "Emp_Id: %d\n", emp.Emp_Id);
        fprintf(payslip_file, "Emp_Name: %s\n", emp.Emp_Name);
        fprintf(payslip_file, "DOB: %d-%d-%d\n", emp.dob.Day, emp.dob.Month, emp.dob.Year);
        fprintf(payslip_file, "Age: %d\n", emp.Age);
        fprintf(payslip_file, "Address: %s, %s, %s, %s, %d\n", emp.address.Door_No, emp.address.Street, emp.address.Area, emp.address.City, emp.address.Pincode);
        fprintf(payslip_file, "Dept: %s\n", emp.Dept);
        fprintf(payslip_file, "Basic_Salary: %.2f\n", emp.Basic_Salary);
```

```c
        fprintf(payslip_file, "Dept: %s\n", emp.Dept);
        fprintf(payslip_file, "Basic_Salary: %.2f\n", emp.Basic_Salary);
        fprintf(payslip_file, "Dearness_Allowance: %.2f\n", emp.Allowance[0]);
        fprintf(payslip_file, "HRA: %.2f\n", emp.Allowance[1]);
        fprintf(payslip_file, "Medical_Allowance: %.2f\n", emp.Allowance[2]);
        fprintf(payslip_file, "Gross_Salary: %.2f\n", emp.Gross_Salary);
        fprintf(payslip_file, "PF: %.2f\n", emp.Deduction[0]);
        fprintf(payslip_file, "Income_Tax: %.2f\n", emp.Deduction[1]);
        fprintf(payslip_file, "Net_Salary: %.2f\n", emp.Net_Salary);

        fclose(payslip_file);
    }

    fclose(file);
}

int main() {
    const char* filename = "employee_data.dat";

    createEmployeeDa  const char *filename
    computeSalaries(filename);

    return 0;
}
```

```
Employee Payslip
----------------
Emp_Id: 9876
Emp_Name: rahul
DOB: 16-7-1978
Age: 45
Address: 75, 6, wer, nik, 914823
Dept: chem
Basic_Salary: 75000.00
Dearness_Allowance: 31500.00
HRA: 7500.00
Medical_Allowance: 11250.00
Gross_Salary: 125250.00
PF: 9000.00
Income_Tax: 25050.00
Net_Salary: 91200.00
```

```
Employee Payslip
----------------
Emp_Id: 983
Emp_Name: suresh
DOB: 18-9-1986
Age: 37
Address: 67, 8, nki, wef, 192305
Dept: mech
Basic_Salary: 75000.00
Dearness_Allowance: 31500.00
HRA: 7500.00
Medical_Allowance: 11250.00
Gross_Salary: 125250.00
PF: 9000.00
Income_Tax: 25050.00
Net_Salary: 91200.00
```

```
Employee Payslip
----------------
Emp_Id: 1234
Emp_Name: shiv
DOB: 11-2-1989
Age: 35
Address: 4, 7, sdf, vbn, 400987
Dept: it
Basic_Salary: 150000.00
Dearness_Allowance: 63000.00
HRA: 15000.00
Medical_Allowance: 22500.00
Gross_Salary: 250500.00
PF: 18000.00
Income_Tax: 50100.00
Net_Salary: 182400.00
```

```
Employee Payslip
----------------
Emp_Id: 2345
Emp_Name: ravi
DOB: 13-6-1978
Age: 45
Address: 78, 3, oiv, akfd, 827345
Dept: tech
Basic_Salary: 85000.00
Dearness_Allowance: 35700.00
HRA: 8500.00
Medical_Allowance: 12750.00
Gross_Salary: 141950.00
PF: 10200.00
Income_Tax: 28390.00
Net_Salary: 103360.00
```

```
Employee Payslip
----------------
Emp_Id: 4567
Emp_Name: ram
DOB: 12-6-1998
Age: 25
Address: 8, 9, sdf, cvb, 123456
Dept: comm
Basic_Salary: 100000.00
Dearness_Allowance: 42000.00
HRA: 10000.00
Medical_Allowance: 15000.00
Gross_Salary: 167000.00
PF: 12000.00
Income_Tax: 33400.00
Net_Salary: 121600.00
```

Write a C program to input multiple lines of text and to determine the number of vowels, consonants, digits, whitespace characters and other characters for each line and finally to find the average number of vowels per line, consonants per line etc. Store the multiple lines of text whose maximum length is unspecified. Maintain a pointer to each string within a one-dimensional array of pointers.

Function prototype:

→ void countCharacters (char * line, int * vowels, int * consonants, int *digits, int * whitespace, int * others);

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

#define MAX_LINES 1000

void countCharacters(char *line, int *vowels, int *consonants, int *digits, int *whitespaces, int *others) {
    *vowels = *consonants = *digits = *whitespaces = *others = 0;
    char ch;
    while ((ch = *line++) != '\0') {
        if (isalpha(ch)) {
            ch = tolower(ch);
            if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                (*vowels)++;
            } else {
                (*consonants)++;
            }
        } else if (isdigit(ch)) {
            (*digits)++;
        } else if (isspace(ch)) {
            (*whitespaces)++;
        } else {
            (*others)++;
        }
    }
}

int main() {
    char *lines[MAX_LINES];
    char buffer[1024];
    int line_count = 0;
    int total_vowels = 0, total_consonants = 0, total_digits = 0, total_whitespaces = 0, total_others = 0;

    printf("Enter lines of text (end with an empty line):\n");
    while (fgets(buffer, sizeof(buffer), stdin) != NULL && buffer[0] != '\n') {
        lines[line_count] = (char *)malloc(strlen(buffer) + 1);
        if (lines[line_count] == NULL) {
            perror("Unable to allocate memory");
            exit(EXIT_FAILURE);
        }
        strcpy(lines[line_count], buffer);
        line_count++;
    }
```

```c
    }

    int vowels, consonants, digits, whitespaces, others;
    for (int i = 0; i < line_count; i++) {
        countCharacters(lines[i], &vowels, &consonants, &digits, &whitespaces, &others);
        printf("Line %d: Vowels = %d, Consonants = %d, Digits = %d, Whitespaces = %d, Others = %d\n",
                i + 1, vowels, consonants, digits, whitespaces, others);
        total_vowels += vowels;
        total_consonants += consonants;
        total_digits += digits;
        total_whitespaces += whitespaces;
        total_others += others;
    }

    printf("\nAverages per line:\n");
    printf("Average Vowels per line: %.2f\n", (float)total_vowels / line_count);
    printf("Average Consonants per line: %.2f\n", (float)total_consonants / line_count);
    printf("Average Digits per line: %.2f\n", (float)total_digits / line_count);
    printf("Average Whitespaces per line: %.2f\n", (float)total_whitespaces / line_count);
    printf("Average Others per line: %.2f\n", (float)total_others / line_count);

    for (int i = 0; i < line_count; i++) {
        free(lines[i]);
    }

    return 0;
}
```

```
Enter lines of text (end with an empty line):
hi, how are you? i am fine. looks like it's a nice day today!

Line 1: Vowels = 21, Consonants = 22, Digits = 0, Whitespaces = 14, Others = 5

Averages per line:
Average Vowels per line: 21.00
Average Consonants per line: 22.00
Average Digits per line: 0.00
Average Whitespaces per line: 14.00
Average Others per line: 5.00
(base) shivaaneesk@Shivaanees-MacBook-Air C Lab %
```

Write an interactive C program to maintain a list of names, addresses and telephone numbers. Store the information as records in a file by representing each record as a structure. Perform the following operations: i) Add a new record at the end of file ii) Retrieve and display the entire record for a given name iii) List all names with their addresses and telephone numbers. Note: Use fscanf and fprintf functions for reading and writing to the file

Structures used:

```
→ typedef struct
  {
      char name[50];
      char address[100];
      char phone[15];
  }Record.
```

→ Functions Used:

→ void addRecord();

→ void retriereRecord();

→ void listRecords();

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

#define FILENAME "records.txt"

typedef struct {
    char name[50];
    char address[100];
    char phone[15];
} Record;

void addRecord() {
    FILE *file = fopen(FILENAME, "a");
    if (!file) {
        printf("Unable to open file for appending!\n");
        return;
    }

    Record newRecord;

    printf("Enter Name: ");
    scanf(" %[^\n]", newRecord.name);

    printf("Enter Address: ");
    scanf(" %[^\n]", newRecord.address);

    printf("Enter Telephone Number: ");
    scanf(" %[^\n]", newRecord.phone);

    fprintf(file, "%s|%s|%s\n", newRecord.name, newRecord.address, newRecord.phone);

    fclose(file);
    printf("Record added successfully!\n");
}

void retrieveRecord() {
    char searchName[50];
    printf("Enter the name to search: ");
    scanf(" %[^\n]", searchName);

    FILE *file = fopen(FILENAME, "r");
    if (!file) {
```

```c
        printf("Unable to open file for reading!\n");
        return;
    }

    Record record;
    int found = 0;

    while (fscanf(file, " %[^|]|%[^|]|%[^\n]\n", record.name, record.address, record.phone) != EOF) {
        if (strcmp(record.name, searchName) == 0) {
            printf("Record Found:\n");
            printf("Name: %s\n", record.name);
            printf("Address: %s\n", record.address);
            printf("Telephone Number: %s\n", record.phone);
            found = 1;
            break;
        }
    }

    fclose(file);

    if (!found) {
        printf("No record found for the name %s.\n", searchName);
    }
}

void listRecords() {
    FILE *file = fopen(FILENAME, "r");
    if (!file) {
        printf("Unable to open file for reading!\n");
        return;
    }

    Record record;

    printf("Listing all records:\n");

    while (fscanf(file, " %[^|]|%[^|]|%[^\n]\n", record.name, record.address, record.phone) != EOF) {
        printf("Name: %s\n", record.name);
        printf("Address: %s\n", record.address);
        printf("Telephone Number: %s\n", record.phone);
        printf("\n");
```

```c
        printf("\n");
    }

    fclose(file);
}

int main() {
    int choice;

    while (1) {
        printf("\nPhonebook Menu:\n");
        printf("1. Add Record\n");
        printf("2. Retrieve Record\n");
        printf("3. List All Records\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addRecord();
                break;
            case 2:
                retrieveRecord();
                break;
            case 3:
                listRecords();
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice, please try again.\n");
        }
    }

    return 0;
}
```

```
Phonebook Menu:
1. Add Record
2. Retrieve Record
3. List All Records
4. Exit
Enter your choice: 1
Enter Name: shivaanee
Enter Address: anna nagar
Enter Telephone Number: 1234567890
Record added successfully!

Phonebook Menu:
1. Add Record
2. Retrieve Record
3. List All Records
4. Exit
Enter your choice: 2
Enter the name to search: shivaanee
Record Found:
Name: shivaanee
Address: anna nagar
Telephone Number: 1234567890

Phonebook Menu:
1. Add Record
2. Retrieve Record
3. List All Records
4. Exit
Enter your choice: 3
Listing all records:
Name: shivaanee
Address: anna nagar
Telephone Number: 1234567890


Phonebook Menu:
1. Add Record
2. Retrieve Record
3. List All Records
4. Exit
Enter your choice: 4
Exiting...
```

Modify 2 by using fread and fwrite functions for reading and writing. Perform the following operations: i) Insert a new record in m th position ii) Delete a record based upon the given name iii) Display n th record

Structures used:

Same as Q 2

Functions used:

```
→ void inputEmployeeData (Employee *emp);
→ void createEmployeeDatabase (Employee employees[], int num_employees);
→ void loadEmployeeData (Employee employees[], int * num_employees);
→ int calculateAge (DOB dob, DOB current_date);
→ void calculateAllowances (Employee *emp);
→ void calculateGrossSalary (Employee *emp);
→ void calculateDeductions (Employee * emp);
→ void calculateNetSalary (Employee *emp);
→ void generatePaySlips (Employee employees[], int num_employees);
→ void insertEmployee (Employee employees[], int * num_employees,
                       int position);
→ void deleteEmployee (Employee employees[], int * num_employees,
                       char * name);
→ void displayNthRecord (Employee employees[], int num_employees,
                         int n);
```

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

#define MAX_EMPLOYEES 5
#define FILENAME "employee_data.dat"

typedef struct {
    int Day;
    int Month;
    int Year;
} DOB;

typedef struct {
    char Door_No[10];
    char Street[50];
    char Area[50];
    char City[50];
    char Pincode[10];
} Address;

typedef struct {
    int Emp_Id;
    char Emp_Name[50];
    DOB dob;
    int Age;
    Address address;
    char Dept[30];
    float Basic_Salary;
    float Allowance[3]; // 0: Dearness_Allowance, 1: HRA, 2: Medical_Allowance
    float Deduction[2]; // 0: PF, 1: Income_Tax
    float Gross_Salary;
    float Net_Salary;
} Employee;

void inputEmployeeData(Employee *emp) {
    printf("Enter Employee ID: ");
    scanf("%d", &emp->Emp_Id);
    printf("Enter Employee Name: ");
    scanf(" %[^\n]", emp->Emp_Name);
    printf("Enter DOB (DD MM YYYY): ");
    scanf("%d %d %d", &emp->dob.Day, &emp->dob.Month, &emp->dob.Year);
```

```c
    printf("Enter Address - Door No: ");
    scanf(" %[^\n]", emp->address.Door_No);
    printf("Enter Street: ");
    scanf(" %[^\n]", emp->address.Street);
    printf("Enter Area: ");
    scanf(" %[^\n]", emp->address.Area);
    printf("Enter City: ");
    scanf(" %[^\n]", emp->address.City);
    printf("Enter Pincode: ");
    scanf(" %[^\n]", emp->address.Pincode);
    printf("Enter Department: ");
    scanf(" %[^\n]", emp->Dept);
    printf("Enter Basic Salary: ");
    scanf("%f", &emp->Basic_Salary);
}

void createEmployeeDatabase(Employee employees[], int num_employees) {
    FILE *file = fopen(FILENAME, "wb");
    if (!file) {
        printf("Error opening file for writing.\n");
        return;
    }

    for (int i = 0; i < num_employees; i++) {
        printf("\nEnter details for Employee %d\n", i + 1);
        inputEmployeeData(&employees[i]);
    }

    fwrite(employees, sizeof(Employee), num_employees, file);

    fclose(file);
}

void loadEmployeeData(Employee employees[], int *num_employees) {
    FILE *file = fopen(FILENAME, "rb");
    if (!file) {
        printf("Error opening file for reading.\n");
        return;
    }

    *num_employees = fread(employees, sizeof(Employee), MAX_EMPLOYEES, file);
```

```c
    fclose(file);
}

int calculateAge(DOB dob, DOB current_date) {
    int age = current_date.Year - dob.Year;
    if (current_date.Month < dob.Month || (current_date.Month == dob.Month && current_date.Day < dob.Day)) {
        age--;
    }
    return age;
}

void calculateAllowances(Employee *emp) {
    emp->Allowance[0] = 0.42 * emp->Basic_Salary; // Dearness Allowance
    emp->Allowance[1] = 0.10 * emp->Basic_Salary; // HRA
    emp->Allowance[2] = 0.15 * emp->Basic_Salary; // Medical Allowance
}

void calculateGrossSalary(Employee *emp) {
    calculateAllowances(emp);
    emp->Gross_Salary = emp->Basic_Salary + emp->Allowance[0] + emp->Allowance[1] + emp->Allowance[2];
}

void calculateDeductions(Employee *emp) {
    emp->Deduction[0] = 0.12 * emp->Basic_Salary; // PF
    emp->Deduction[1] = 0.20 * emp->Gross_Salary; // Income Tax
}

void calculateNetSalary(Employee *emp) {
    calculateDeductions(emp);
    emp->Net_Salary = emp->Gross_Salary - emp->Deduction[0] - emp->Deduction[1];
}

void generatePayslips(Employee employees[], int num_employees) {
    for (int i = 0; i < num_employees; i++) {
        char filename[20];
        sprintf(filename, "payslip_%d.txt", employees[i].Emp_Id);
        FILE *file = fopen(filename, "w");
        if (!file) {
            printf("Error opening file for writing.\n");
            continue;
        }
```

```c
        fprintf(file, "Payslip for Employee ID %d\n", employees[i].Emp_Id);
        fprintf(file, "Name: %s\n", employees[i].Emp_Name);
        fprintf(file, "Department: %s\n", employees[i].Dept);
        fprintf(file, "Basic Salary: %.2f\n", employees[i].Basic_Salary);
        fprintf(file, "Gross Salary: %.2f\n", employees[i].Gross_Salary);
        fprintf(file, "Net Salary: %.2f\n", employees[i].Net_Salary);

        fclose(file);
    }
}

void insertEmployee(Employee employees[], int *num_employees, int position) {
    if (*num_employees >= MAX_EMPLOYEES || position > *num_employees) {
        printf("Cannot insert employee at the given position.\n");
        return;
    }

    for (int i = *num_employees; i > position; i--) {
        employees[i] = employees[i - 1];
    }

    printf("\nEnter details for new Employee\n");
    inputEmployeeData(&employees[position]);

    (*num_employees)++;

    FILE *file = fopen(FILENAME, "wb");
    if (!file) {
        printf("Error opening file for writing.\n");
        return;
    }

    fwrite(employees, sizeof(Employee), *num_employees, file);

    fclose(file);
}

void deleteEmployee(Employee employees[], int *num_employees, char *name) {
    int found = 0;

    for (int i = 0; i < *num_employees; i++) {
```

```c
        if (strcmp(employees[i].Emp_Name, name) == 0) {
            found = 1;
            for (int j = i; j < *num_employees - 1; j++) {
                employees[j] = employees[j + 1];
            }
            (*num_employees)--;
            break;
        }
    }

    if (!found) {
        printf("Employee with name %s not found.\n", name);
        return;
    }

    FILE *file = fopen(FILENAME, "wb");
    if (!file) {
        printf("Error opening file for writing.\n");
        return;
    }

    fwrite(employees, sizeof(Employee), *num_employees, file);

    fclose(file);
}

void displayNthRecord(Employee employees[], int num_employees, int n) {
    if (n >= num_employees || n < 0) {
        printf("Invalid record number.\n");
        return;
    }

    printf("\nDetails of Employee %d\n", n + 1);
    printf("Employee ID: %d\n", employees[n].Emp_Id);
    printf("Name: %s\n", employees[n].Emp_Name);
    printf("DOB: %d/%d/%d\n", employees[n].dob.Day, employees[n].dob.Month, employees[n].dob.Year);
    printf("Address: %s, %s, %s, %s, %s\n", employees[n].address.Door_No, employees[n].address.Street,
        employees[n].address.Area, employees[n].address.City, employees[n].address.Pincode);
    printf("Department: %s\n", employees[n].Dept);
    printf("Basic Salary: %.2f\n", employees[n].Basic_Salary);
}
```

```c
int main() {
    Employee employees[MAX_EMPLOYEES];
    int num_employees;

    createEmployeeDatabase(employees, MAX_EMPLOYEES);
    loadEmployeeData(employees, &num_employees);

    DOB current_date;
    printf("Enter current date (DD MM YYYY): ");
    scanf("%d %d %d", &current_date.Day, &current_date.Month, &current_date.Year);

    for (int i = 0; i < num_employees; i++) {
        employees[i].Age = calculateAge(employees[i].dob, current_date);
        calculateGrossSalary(&employees[i]);
        calculateNetSalary(&employees[i]);
    }

    generatePayslips(employees, num_employees);

    int choice, position, n;
    char name[50];

    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert New Employee\n");
        printf("2. Delete Employee by Name\n");
        printf("3. Display Nth Employee\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter position to insert new employee (0 to %d): ", num_employees);
                scanf("%d", &position);
                insertEmployee(employees, &num_employees, position);
                break;
            case 2:
                printf("Enter name of the employee to delete: ");
                scanf(" %[^\n]", name);
                deleteEmployee(employees, &num_employees, name);
```

```c
                break;
            case 3:
                printf("Enter record number to display (0 to %d): ", num_employees - 1);
                scanf("%d", &n);
                displayNthRecord(employees, num_employees, n);
                break;
            case 4:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

```
Enter current date (DD MM YYYY): 05 06 2024

Menu:
1. Insert New Employee
2. Delete Employee by Name
3. Display Nth Employee
4. Exit
Enter your choice: 1
Enter position to insert new employee (0 to 5): 3
Cannot insert employee at the given position.

Menu:
1. Insert New Employee
2. Delete Employee by Name
3. Display Nth Employee
4. Exit
Enter your choice: 2
Enter name of the employee to delete: shiv

Menu:
1. Insert New Employee
2. Delete Employee by Name
3. Display Nth Employee
4. Exit
Enter your choice: 3
Enter record number to display (0 to 3): 2

Details of Employee 3
Employee ID: 9823
Name: ram
DOB: 30/1/1987
Address: 34, 8, pqs, dyh, 103429
Department: tech
Basic Salary: 95000.00

Menu:
1. Insert New Employee
2. Delete Employee by Name
3. Display Nth Employee
4. Exit
Enter your choice: 4
```