/\* Implement Complex numbers using structures. Write functions to add, multiply, subtract two complex numbers. \*/

#include <stdio.h>

// Define a structure for complex numbers

struct Complex {

double real;

double imaginary;

};

// Function to add two complex numbers

struct Complex addComplex(const struct Complex\* c1, const struct Complex\* c2) {

struct Complex result;

result.real = c1->real + c2->real;

result.imaginary = c1->imaginary + c2->imaginary;

return result;

}

// Function to subtract two complex numbers

struct Complex subtractComplex(const struct Complex\* c1, const struct Complex\* c2) {

struct Complex result;

result.real = c1->real - c2->real;

result.imaginary = c1->imaginary - c2->imaginary;

return result;

}

// Function to multiply two complex numbers

struct Complex multiplyComplex(const struct Complex\* c1, const struct Complex\* c2) {

struct Complex result;

result.real = c1->real \* c2->real - c1->imaginary \* c2->imaginary;

result.imaginary = c1->real \* c2->imaginary + c1->imaginary \* c2->real;

return result;

}

int main() {

struct Complex num1 = {3.0, 4.0};

struct Complex num2 = {1.0, 2.0};

struct Complex sum = addComplex(&num1, &num2);

struct Complex difference = subtractComplex(&num1, &num2);

struct Complex product = multiplyComplex(&num1, &num2);

printf("Sum: %.2f + %.2fi\n", sum.real, sum.imaginary);

printf("Difference: %.2f + %.2fi\n", difference.real, difference.imaginary);

printf("Product: %.2f + %.2fi\n", product.real, product.imaginary);

return 0;

}

/\* Write a C program to implement the following functions. Use pointers and

dynamic memory management functions.

i. To read one Student object where Student is a structure with name, roll number and CGPA as the data members

ii. To display one Student object

iii. To sort an array of Student structures according to the roll number. \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Define the Student structure

struct Student {

char name[50];

int rollNumber;

float cgpa;

};

// Function to read a Student object

void readStudent(struct Student \*student) {

printf("Enter student name: ");

scanf("%s", student->name);

printf("Enter roll number: ");

scanf("%d", &(student->rollNumber));

printf("Enter CGPA: ");

scanf("%f", &(student->cgpa));

}

// Function to display a Student object

void displayStudent(const struct Student \*student) {

printf("Name: %s\n", student->name);

printf("Roll Number: %d\n", student->rollNumber);

printf("CGPA: %.2f\n", student->cgpa);

}

// Function to sort an array of Student structures using bubble sort

void bubbleSort(struct Student \*students, int n) {

for (int i = 0; i < n - 1; i++) {

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

// Compare roll numbers and swap if necessary

if (students[j].rollNumber > students[j + 1].rollNumber) {

struct Student temp = students[j];

students[j] = students[j + 1];

students[j + 1] = temp;

}

}

}

}

int main() {

int n;

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

scanf("%d", &n);

// Allocate memory for an array of Student objects

struct Student \*students = (struct Student \*)malloc(n \* sizeof(struct Student));

if (students == NULL) {

printf("Memory allocation failed\n");

return 1;

}

// Read student data

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

printf("\nEnter details for student %d:\n", i + 1);

readStudent(&students[i]);

}

// Sort the array of Student structures based on roll number using bubble sort

bubbleSort(students, n);

// Display student data

printf("\nStudent List (Sorted by Roll Number):\n");

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

printf("\nStudent %d:\n", i + 1);

displayStudent(&students[i]);

}

// Free dynamically allocated memory

free(students);

return 0;

}

/\* Samuel wants to store the data of his employees, which includes the following fields:

(1) Name of the emplovee (11) Date of birth which is a collection of dav. month. vear!

(in) Address which is a collection of house number. zip code and state!. Write a 'C

program to read and display the data of N emplovees using pointers to array of structures \*/

#include <stdio.h>

#include <stdlib.h>

struct Date {

int day;

int month;

int year;

};

struct Address {

char houseNumber[20];

char zipCode[20];

char state[30];

};

struct Employee {

char name[50];

struct Date dob;

struct Address address;

};

void readEmployee(struct Employee \*employee) {

printf("Enter employee name: ");

scanf("%s", employee->name);

printf("Enter date of birth (dd mm yyyy): ");

scanf("%d %d %d", &employee->dob.day, &employee->dob.month, &employee->dob.year);

printf("Enter house number: ");

scanf("%s", employee->address.houseNumber);

printf("Enter zip code: ");

scanf("%s", employee->address.zipCode);

printf("Enter state: ");

scanf("%s", employee->address.state);

}

void displayEmployee(const struct Employee \*employee) {

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

printf("Date of Birth: %d/%d/%d\n", employee->dob.day, employee->dob.month, employee->dob.year);

printf("Address: %s, %s, %s\n", employee->address.houseNumber, employee->address.zipCode, employee->address.state);

}

int main() {

int numEmployees;

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

scanf("%d", &numEmployees);

struct Employee \*employees = (struct Employee \*)malloc(numEmployees \* sizeof(struct Employee));

if (employees == NULL) {

printf("Memory allocation failed\n");

return 1;

}

for (int i = 0; i < numEmployees; ++i) {

printf("Enter details for employee %d:\n", i + 1);

readEmployee(&employees[i]);

}

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

for (int i = 0; i < numEmployees; ++i) {

printf("\nEmployee %d:\n", i + 1);

displayEmployee(&employees[i]);

}

free(employees);

return 0;

}

/\* Create a structure STUDENT consisting of variables of structures:

i. DOB {day, month (use pointer), year). ii. STU\_INFO (reg\_no, name(use pointer), address),

i. COLLEGE (college\_name (use pointer), university\_name )

where structure types from i to in are declared outside the STUDENT independently.

Show how to read and display member variables of DOB type if pointer variable is created for

DOB inside STUDENT and STUDENT variable is also a pointer variable. The program should read

and display the values of all members of STUDENT structure. \*/

#include <stdio.h>

#include <stdlib.h>

// Structure for Date of Birth (DOB)

struct DOB {

int day;

int \*month; // Using pointer for month

int year;

};

// Structure for Student Information (STU\_INFO)

struct STU\_INFO {

int reg\_no;

char \*name; // Using pointer for name

char \*address;

};

// Structure for College

struct COLLEGE {

char \*college\_name;

char \*university\_name;

};

// Structure for Student (containing nested structures)

struct STUDENT {

struct DOB dob;

struct STU\_INFO stu\_info;

struct COLLEGE college;

};

int main() {

// Declare a pointer to STUDENT structure

struct STUDENT \*student;

// Allocate memory for student structure

student = (struct STUDENT \*)malloc(sizeof(struct STUDENT));

if (student == NULL) {

printf("Memory allocation failed\n");

return 1;

}

// Allocate memory for nested structures

student->dob.month = (int \*)malloc(sizeof(int));

student->stu\_info.name = (char \*)malloc(50 \* sizeof(char));

student->stu\_info.address = (char \*)malloc(100 \* sizeof(char));

student->college.college\_name = (char \*)malloc(50 \* sizeof(char));

student->college.university\_name = (char \*)malloc(50 \* sizeof(char));

// Read student details

printf("Enter student's date of birth (day month year): ");

scanf("%d %d %d", &student->dob.day, student->dob.month, &student->dob.year);

printf("Enter student's registration number: ");

scanf("%d", &student->stu\_info.reg\_no);

printf("Enter student's name: ");

scanf("%s", student->stu\_info.name);

printf("Enter student's address: ");

scanf(" %s [^\n]", student->stu\_info.address);

printf("Enter college name: ");

scanf("%s", student->college.college\_name);

printf("Enter university name: ");

scanf("%s", student->college.university\_name);

printf("\n \n ");

// Display student details

printf("\nStudent details:\n");

printf("Date of Birth: %d/%d/%d\n", student->dob.day, \*(student->dob.month), student->dob.year);

printf("Registration Number: %d\n", student->stu\_info.reg\_no);

printf("Name: %s\n", student->stu\_info.name);

printf("Address: %s\n", student->stu\_info.address);

printf("College: %s\n", student->college.college\_name);

printf("University: %s\n", student->college.university\_name);

// Free allocated memory

free(student->dob.month);

free(student->stu\_info.name);

free(student->stu\_info.address);

free(student->college.college\_name);

free(student->college.university\_name);

free(student);

return 0;

}