Academic year 2023-2024 (Even Semester)

CIE I: Scheme and Solutions

Course	**Principles of Programming using C**		
Date	June 2024	Maximum Marks	50
Course Code	CS222AI	Duration	90 Min
Sem	II	CIE – II	

SL No	Answers	M	BT	CO

1.a	#include <stdio.h>
	
	void stringConcatenate(char str1[], char str2[]) {
	int i = 0, j = 0;
	// Find the end of the first string
	while (str1[i] != '\0') {
	i++;
	}
	// Append the second string to the first string
	while (str2[j] != '\0') {
	str1[i] = str2[j];
	i++;
	j++;
	}
	str1[i] = '\0'; // Null terminate the concatenated string
	}
	
	int main() {
	char str1[100], str2[100];
	printf("Enter the first string: ");
	gets(str1);
	printf("Enter the second string: ");
	gets(str2);
	
	stringConcatenate(str1, str2);
	printf("Concatenated string is: %s\n", str1);
	
	return 0;
	}
	Explanation:…3M
	1. The stringConcatenate function takes two character arrays (strings) as input.
	2. It uses an int variable i to find the end of the first string.
	3. Then, it uses another int variable j to traverse the second string and append each character to the end of the first string.
	4. After appending all characters from the second string, it null terminates the concatenated string.
	5. The main function reads two strings from the user, calls the stringConcatenate function, and prints the concatenated string.
		06	L3	CO2

1.b	Explanation of scanf(), gets(),fgets() and getchar() functions ..1*4=4	4	L2	CO1

2.a	#include <stdio.h> // Reading strings…3M, Sorting…3M
	#include <string.h>
	#define MAX_STRINGS 20
	#define MAX_LENGTH 100
	int main() {
	char strings[MAX_STRINGS][MAX_LENGTH];
	char temp[MAX_LENGTH];
	int i, j;
	printf("Enter %d strings:\n", MAX_STRINGS);
	for (i = 0; i < MAX_STRINGS; i++) {
	printf("String %d: ", i + 1);
	fgets(strings[i], MAX_LENGTH, stdin);
		6	L3	CO2

	}
	
	// Sorting strings using bubble sort
	for (i = 0; i < MAX_STRINGS - 1; i++) {
	for (j = i + 1; j < MAX_STRINGS; j++) {
	if (strcmp(strings[i], strings[j]) > 0) {
	strcpy(temp, strings[i]);
	strcpy(strings[i], strings[j]);
	strcpy(strings[j], temp);
	}
	}
	}
	printf("\nStrings in alphabetical order:\n");
	for (i = 0; i < MAX_STRINGS; i++) {
	printf("String %d: %s\n", i + 1, strings[i]);
	}
	return 0;
	}			

2.b	A function in C is a block of code that performs a specific task. It can be called multiple times within a program to execute the task. Functions help in modularizing the code, making it more readable and reusable….1M A **function declaration** (or prototype) specifies the function's name, return type, and parameters but does not include the body of the function. It tells the compiler about the function's existence before its actual definition…1.5M A **function definition** includes the function's name, return type, parameters, and the body (code) of the function. It provides the actual implementation of the function…1.5M	04	L3	CO2

3.a	In C, when passing a multi-dimensional array to a function, the function needs to know the size of all dimensions except the first. This is because the compiler needs to compute the address of elements correctly…1M
	Syntactically correct program..5M
	#include <stdio.h>
	void print2DArray(int arr[][3], int rows) {
	for (int i = 0; i < rows; i++) {
	for (int j = 0; j < 3; j++) {
	if(i==j){
	printf("%d ", arr[i][j]);}
	}
	printf("\n");
	}
	}
	int main() {
	int matrix[2][3] = {
	{1, 2, 3},
	{4, 5, 6}
	};
	
	print2DArray(matrix, 2);
		07	L3	CO3

	return 0; }			

3.b	The const keyword indicates that the array elements cannot be modified within the function. This ensures that the function only reads the array data, preventing accidental changes.
	Example:
	#include <stdio.h>
	void displayArray(const int arr[], int size) {
	for (int i = 0; i < size; i++) {
	printf("%d ", arr[i]);
	}
	printf("\n");
	}
	int main() {
	int numbers[] = {10, 20, 30, 40, 50};
	int size = sizeof(numbers) / sizeof(numbers[0]);
	
	displayArray(numbers, size);
	
	return 0;
	}
		03	L2	CO1

4.a	Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem. It is particularly useful for problems that can be broken down into smaller, similar subproblems. **Base Case:** The condition under which the recursion ends. Without a base case, recursion would lead to an infinite loop. **Recursive Case:** The part of the function where the function calls itself with a modified argument, moving towards the base case … 2M Example: Fibonacci Sequence The Fibonacci sequence is defined as:….1M Fib(0) = 0 Fib(1) = 1 Fib(n) = Fib(n-1) + Fib(n-2) for n > 1 Syntactically correct program..3M	06	L3	CO3
4.b	**a) Formal parameters** are part of the function definition, defining the types and names of the values that the function expects to receive. **Actual parameters** are part of the function call, providing the specific values to be passed to the function's formal parameters. b) **Scope** determines where in the program a variable can be accessed and used. **Lifetime** determines how long a variable exists in memory, from its	04	L3	CO4

	creation to its destruction			

5.a	#define MAX_HOTELS 5
	#define MAX_CUSTOMERS 10
	
	// Structure to store hotel information
	struct Hotel {
	char H_Name[50];
	char H_City[50];
	int Number_of_rooms;
	float Room_charges;
	};
	// Structure to store customer information
	struct Customer {
	char C_Name[50];
	char C_Address[100];
	char Phone_no[15];
	char DOB[11]; // Format: YYYY-MM-DD
	};
	// Structure to store reservation details
	struct Reservation {
	int Reservation_ID;
	struct Customer customer;
	struct Hotel hotel;
	int Number_of_nights;
	};
	// Structure to store payment details
	struct Payment {
	int Payment_ID;
	struct Reservation reservation;
	float Amount_paid;
	}; …. 4M
	// Function to display the list of hotels in a particular city
	void displayHotelsInCity(struct Hotel hotels[], int hotelCount, const char* city) {
	printf("Hotels in %s:\n", city);
	for (int i = 0; i < hotelCount; i++) {
	if (strcmp(hotels[i].H_City, city) == 0) {
	printf("Hotel Name: %s\n", hotels[i].H_Name);
	printf("City: %s\n", hotels[i].H_City);
	printf("Number of Rooms: %d\n", hotels[i].Number_of_rooms);
	printf("Room Charges: %.2f\n", hotels[i].Room_charges);
		03	L2	CO2

```
            printf("-------------------------\n");
        }
    }
}
// Function to print reservation details
void printReservationDetails(struct Reservation reservation) {
    printf("Reservation ID: %d\n", reservation.Reservation_ID);
    printf("Customer Name: %s\n", reservation.customer.C_Name);
    printf("Customer Address: %s\n", reservation.customer.C_Address);
    printf("Customer Phone: %s\n", reservation.customer.Phone_no);
    printf("Customer DOB: %s\n", reservation.customer.DOB);
    printf("Hotel Name: %s\n", reservation.hotel.H_Name);
    printf("Hotel City: %s\n", reservation.hotel.H_City);
    printf("Number of Rooms: %d\n",
reservation.hotel.Number_of_rooms);
    printf("Room Charges: %.2f\n", reservation.hotel.Room_charges);
    printf("Number of Nights: %d\n", reservation.Number_of_nights);
    printf("Total Charge: %.2f\n", reservation.Number_of_nights *
reservation.hotel.Room_charges);
    printf("-------------------------\n");
}
// Function to print payment receipt
void printPaymentReceipt(struct Payment payment) {
    printf("Payment ID: %d\n", payment.Payment_ID);
    printf("Amount Paid: %.2f\n", payment.Amount_paid);
    printf("Reservation Details:\n");
    printReservationDetails(payment.reservation);
}
int main() {
    // Array of hotels
    struct Hotel hotels[MAX_HOTELS] = {
        {"Hotel A", "City X", 50, 100.0},
        {"Hotel B", "City Y", 70, 150.0},
        {"Hotel C", "City X", 40, 120.0},
        {"Hotel D", "City Z", 60, 130.0},
        {"Hotel E", "City Y", 80, 140.0}
    };

    // Array of reservations
    struct Reservation reservations[MAX_CUSTOMERS];
    int reservationCount = 0;

    // Array of payments
```

```
struct Payment payments[MAX_CUSTOMERS];
int paymentCount = 0;

// Main menu
int choice;
while (1) {
    printf("\nHotel Management System\n");
    printf("1. Display list of hotels in a particular city\n");
    printf("2. Book a room\n");
    printf("3. Print reservation details\n");
    printf("4. Print payment receipt\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        // Display list of hotels in a particular city
        char city[50];
        printf("Enter city name: ");
        scanf(" %[^\n]%*c", city);
        displayHotelsInCity(hotels, MAX_HOTELS, city);
    } else if (choice == 2) {
        // Book a room
        if (reservationCount >= MAX_CUSTOMERS) {
            printf("No more reservations can be made.\n");
            continue;
        }
        struct Reservation newReservation;
        newReservation.Reservation_ID = reservationCount + 1;
        printf("Enter customer name: ");
        scanf(" %[^\n]%*c", newReservation.customer.C_Name);
        printf("Enter customer address: ");
        scanf(" %[^\n]%*c", newReservation.customer.C_Address);
        printf("Enter customer phone number: ");
        scanf(" %[^\n]%*c", newReservation.customer.Phone_no);
        printf("Enter customer DOB (YYYY-MM-DD): ");
        scanf(" %[^\n]%*c", newReservation.customer.DOB);

        char hotelName[50];
        printf("Enter hotel name: ");
        scanf(" %[^\n]%*c", hotelName);
        int hotelFound = 0;
        for (int i = 0; i < MAX_HOTELS; i++) {
```

```
            if (strcmp(hotels[i].H_Name, hotelName) == 0) {
                newReservation.hotel = hotels[i];
                hotelFound = 1;
                break;
            }
        }
        if (!hotelFound) {
            printf("Hotel not found.\n");
            continue;
        }

        printf("Enter number of nights: ");
        scanf("%d", &newReservation.Number_of_nights);

        reservations[reservationCount++] = newReservation;
        printf("Room booked successfully!\n");
    } else if (choice == 3) {
        // Print reservation details
        int reservationID;
        printf("Enter reservation ID: ");
        scanf("%d", &reservationID);
        int found = 0;
        for (int i = 0; i < reservationCount; i++) {
            if (reservations[i].Reservation_ID == reservationID) {
                printReservationDetails(reservations[i]);
                found = 1;
                break;
            }
        }
        if (!found) {
            printf("Reservation ID not found.\n");
        }
    } else if (choice == 4) {
        // Print payment receipt
        if (paymentCount >= MAX_CUSTOMERS) {
            printf("No more payments can be processed.\n");
            continue;
        }
        struct Payment newPayment;
        newPayment.Payment_ID = paymentCount + 1;

        int reservationID;
        printf("Enter reservation ID for payment: ");
```

```
            scanf("%d", &reservationID);
            int found = 0;
            for (int i = 0; i < reservationCount; i++) {
                if (reservations[i].Reservation_ID == reservationID) {
                    newPayment.reservation = reservations[i];
                    found = 1;
                    break;
                }
            }
            if (!found) {
                printf("Reservation ID not found.\n");
                continue;
            }

            printf("Enter amount paid: ");
            scanf("%f", &newPayment.Amount_paid);

            payments[paymentCount++] = newPayment;
            printf("Payment processed successfully!\n");
        } else if (choice == 5) {
            // Exit the program
            break;
        } else {
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```