1. **ARRAYS**

#include <stdio.h>

#include <string.h>

#define MAX\_TASKS 3

#define DAYS 7

// Structure for each day's tasks

typedef struct {

char dayName[10];

char tasks[MAX\_TASKS][100];

int taskCount;

} Day;

// Initialize days of the week

void initDays(Day week[]) {

const char\* names[] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};

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

strcpy(week[i].dayName, names[i]);

week[i].taskCount = 0;

}

}

// Allow user to input tasks

void addTasks(Day week[]) {

char day[10];

int found = 0;

printf("Enter the day to add tasks (e.g., Monday): ");

scanf("%s", day);

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

if (strcasecmp(week[i].dayName, day) == 0) {

found = 1;

int remaining = MAX\_TASKS - week[i].taskCount;

if (remaining == 0) {

printf("Maximum tasks already added for %s.\n", week[i].dayName);

return;

}

for (int j = week[i].taskCount; j < MAX\_TASKS; j++) {

printf("Enter task %d for %s: ", j + 1, week[i].dayName);

scanf(" %[^\n]", week[i].tasks[j]);

week[i].taskCount++;

char ch;

printf("Add another task for %s? (y/n): ", week[i].dayName);

scanf(" %c", &ch);

if (ch != 'y' && ch != 'Y') break;

}

}

}

if (!found) printf("Invalid day name entered.\n");

}

// Display all tasks

void displayTasks(Day week[]) {

printf("\n--- Weekly Task Calendar ---\n");

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

printf("%s:\n", week[i].dayName);

for (int j = 0; j < week[i].taskCount; j++) {

printf(" - %s\n", week[i].tasks[j]);

}

if (week[i].taskCount == 0) {

printf(" (No tasks)\n");

}

}

}

int main() {

Day week[DAYS];

initDays(week);

int choice;

do {

printf("\n1. Add Tasks\n2. Display Tasks\n3. Exit\nEnter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: addTasks(week); break;

case 2: displayTasks(week); break;

}

} while (choice != 3);

return 0;

}

**2. POINTERS**

#include <stdio.h>

void rearrangeEvenOdd(int\* arr, int size) {

int temp[size];

int\* tempPtr = temp;

int\* ptr = arr;

// Copy evens

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

if (\*(ptr + i) % 2 == 0) {

\*tempPtr++ = \*(ptr + i);

}

}

// Copy odds

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

if (\*(ptr + i) % 2 != 0) {

\*tempPtr++ = \*(ptr + i);

}

}

// Copy back to original array using pointer

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

\*(arr + i) = temp[i];

}

}

void printArray(int\* arr, int size) {

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

printf("%d ", \*(arr + i));

}

printf("\n");

}

int main() {

int arr[] = {3, 2, 4, 1, 5, 6};

int size = sizeof(arr) / sizeof(arr[0]);

printf("Original array:\n");

printArray(arr, size);

rearrangeEvenOdd(arr, size);

printf("After rearranging:\n");

printArray(arr, size);

return 0;

}

**3.STRUCTURES**

#include <stdio.h>

#include <stdbool.h>

// Function to search for a key in a sorted n x n matrix

bool searchMatrix(int matrix[][100], int n, int key) {

int row = 0;

int col = n - 1;

while (row < n && col >= 0) {

if (matrix[row][col] == key) {

return true;

} else if (matrix[row][col] > key) {

col--; // Move left

} else {

row++; // Move down

}

}

return false; // Key not found

}

int main() {

int n, key;

int matrix[100][100];

// Get matrix size

printf("Enter the size of the matrix (n x n): ");

scanf("%d", &n);

// Get matrix elements

printf("Enter the elements of the matrix (each row and column should be sorted):\n");

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

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

printf("Element [%d][%d]: ", i, j);

scanf("%d", &matrix[i][j]);

}

}

// Get the key to search

printf("Enter the key to search: ");

scanf("%d", &key);

// Search and print result

if (searchMatrix(matrix, n, key)) {

printf("Key %d found in the matrix.\n", key);

} else {

printf("Key %d not found in the matrix.\n", key);

}

return 0;

}