ASSIGNMENT 1

1. C program that represents a calendar for a week.

#include <stdio.h>

#include <string.h>

#define DAYS 7

#define MAX\_TASKS 3

#define MAX\_TASK\_LENGTH 100

struct Day {

char dayName[10];

char tasks[MAX\_TASKS][MAX\_TASK\_LENGTH];

};

int main() {

struct Day week[DAYS] = {

{"Monday"}, {"Tuesday"}, {"Wednesday"},

{"Thursday"}, {"Friday"}, {"Saturday"}, {"Sunday"}

};

int choice, taskCount;

char task[MAX\_TASK\_LENGTH];

do {

printf("\nSelect a day to add tasks (1=Monday ... 7=Sunday, 0=Exit): ");

scanf("%d", &choice);

if (choice >= 1 && choice <= 7) {

printf("Enter number of tasks for %s (max %d): ", week[choice-1].dayName, MAX\_TASKS);

scanf("%d", &taskCount);

getchar(); // clear newline

if (taskCount > MAX\_TASKS) taskCount = MAX\_TASKS;

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

printf("Enter task %d: ", i + 1);

fgets(week[choice - 1].tasks[i], MAX\_TASK\_LENGTH, stdin);

week[choice - 1].tasks[i][strcspn(week[choice - 1].tasks[i], "\n")] = '\0';

}

}

} while (choice != 0);

printf("\nWeekly Tasks:\n");

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

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

for (int j = 0; j < MAX\_TASKS; j++) {

if (strlen(week[i].tasks[j]) > 0)

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

}

}

return 0;

}

1. Write a function in C that takes a pointer to an integer array and its size, and then rearranges the array in-place such that all even numbers appear before odd numbers, preserving the original relative order using only pointer arithmetic (no indexing with []).

#include <stdio.h>

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

int temp[size];

int \*evenPtr = temp;

int \*oddPtr = temp;

for (int \*p = arr; p < arr + size; p++) {

if (\*p % 2 == 0) evenPtr++;

}

oddPtr = evenPtr;

for (int \*p = arr; p < arr + size; p++) {

if (\*p % 2 == 0) \*(--evenPtr) = \*p;

}

for (int \*p = arr; p < arr + size; p++) {

if (\*p % 2 != 0) \*(oddPtr++) = \*p;

}

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

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

}

int main() {

int arr[] = {1, 2, 3, 4, 6, 7};

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

rearrangeEvenOdd(arr, size);

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

printf("%d ", arr[i]);

return 0;

}

1. You are given a 2D matrix of size n x n where each row and each column is sorted in increasing order. Write a C function to determine whether a given key exists in the matrix using the most efficient approach. Give me the answers for these questions in a simple method

#include <stdio.h>

#include <stdbool.h>

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

int row = 0, col = n - 1;

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

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

return true;

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

col--;

else

row++;

}

return false;

}

// Example usage

int main() {

int matrix[4][4] = {

{1, 4, 7, 11},

{2, 5, 8, 12},

{3, 6, 9, 16},

{10, 13, 14, 17}

};

int key = 14;

if (searchMatrix(4, matrix, key))

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

else

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

return 0;

}