After using bool isValid(const int arr[],int length, int pos,?) and

void remove\_element(int arr[],int length, int pos), the new updated code is:

#include <stdio.h>

#include <stdbool.h>

#define SIZE 24

#define nRows 8

#define nCols 3

bool isValid(const int arr[], int length, int pos);

bool isValidMatrix(int mat[][nCols], int rows, int cols, int i, int j);

void print\_array(int array[], int length);

void print\_matrix(int mat[][nCols], int rows);

void remove\_element(int arr[], int\* length, int pos);

int main() {

int arr[SIZE] = {0};

int length = SIZE;

int arr2d[nRows][nCols] = {{0}};

// Call to print 1D array

print\_array(arr, length);

remove\_element(arr, &length, 5);

printf("After removing element at index 5:\n");

print\_array(arr, length);

// Call to print 2D matrix

print\_matrix(arr2d, nRows);

return 0;

}

bool isValid(const int arr[], int length, int pos) {

return pos >= 0 && pos < length;

}

bool isValidMatrix(int mat[][nCols], int rows, int cols, int i, int j) {

return i >= 0 && i < rows && j >= 0 && j < cols;

}

void print\_array(int array[], int length) {

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

printf("array[%d] = %d\n", i, array[i]);

}

}

void print\_matrix(int mat[][nCols], int rows) {

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

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

printf("mat[%d][%d] = %d\n", i, j, mat[i][j]);

}

puts("");

}

}

void remove\_element(int arr[], int\* length, int pos) {

// Check if the position is valid

if (!isValid(arr, \*length, pos)) {

printf("Error: Invalid index %d\n", pos);

return; // Exit the function if the index is not valid

}

for (int i = pos; i < \*length - 1; i++) {

arr[i] = arr[i + 1]; // Move element to the left

}

(\*length)--;

arr[\*length] = 0; // Optional: Clear the last element

}