**Assignment 2 Solutions - Arrays and Pointers**

Question 1: Linear Search

#include <stdio.h>

int main() {

int arr[10] = {11, 24, 8, 45, 62, 19, 7, 50, 99, 73};

int size = 10;

int number\_to\_find;

int i;

// Using a 'flag' to know if we found the number

// 0 means not found, 1 means found

int found\_flag = 0;

printf("The array is: ");

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

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

}

printf("\n");

printf("Enter a number to search for: ");

scanf("%d", &number\_to\_find);

// Loop through the array to check each element

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

if (arr[i] == number\_to\_find) {

found\_flag = 1; // Set the flag to 1

break; // Stop the loop since we found it

}

}

// Check the flag after the loop to print the result

if (found\_flag == 1) {

printf("Success! %d is present in the array.\n", number\_to\_find);

} else {

printf("Sorry, %d is not present in the array.\n", number\_to\_find);

}

return 0;

}

**Sample Output:**

**The array is: 11 24 8 45 62 19 7 50 99 73**

**Enter a number to search for: 19**

**Success! 19 is present in the array.**

Question 2: Second Maximum and Second Minimum

#include <stdio.h>

int main() {

int arr[10] = {11, 24, 8, 45, 62, 19, 7, 50, 99, 73};

int size = 10;

int i, j, temp;

// --- Bubble Sort the array ---

// This is a simple way to sort, comparing adjacent elements

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

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

if (arr[j] > arr[j + 1]) {

// Swap the elements

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

printf("The sorted array is: ");

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

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

}

printf("\n");

// After sorting, the second elements are easy to find

printf("Second Minimum element is: %d\n", arr[1]);

printf("Second Maximum element is: %d\n", arr[size - 2]);

return 0;

}

**Sample Output:**

**The sorted array is: 7 8 11 19 24 45 50 62 73 99**

**Second Minimum element is: 8**

**Second Maximum element is: 73**

Question 3: Array Operations (Insert, Delete, Traverse)

#include <stdio.h>

int main() {

int arr[20] = {10, 20, 30, 40, 50};

int size = 5; // Keep track of the current number of elements

int i, position, value;

// --- 1. Traversal ---

printf("Original Array (Traversal):\n");

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

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

}

printf("\n\n");

// --- 2. Insertion ---

printf("--- Insertion ---\n");

position = 2; // Insert at index 2

value = 25;

// Shift elements to the right from the position

for (i = size - 1; i >= position; i--) {

arr[i + 1] = arr[i];

}

arr[position] = value; // Insert the new value

size++; // Increase the size

printf("Array after inserting %d at position %d:\n", value, position);

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

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

}

printf("\n\n");

// --- 3. Deletion ---

printf("--- Deletion ---\n");

position = 4; // Delete element at index 4

// Shift elements to the left from the position

for (i = position; i < size - 1; i++) {

arr[i] = arr[i + 1];

}

size--; // Decrease the size

printf("Array after deleting element at position %d:\n", position);

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

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

}

printf("\n");

return 0;

}

**Sample Output:**

**Original Array (Traversal):**

**10 20 30 40 50**

**--- Insertion ---**

**Array after inserting 25 at position 2:**

**10 20 25 30 40 50**

**--- Deletion ---**

**Array after deleting element at position 4:**

**10 20 25 30 50**

Question 4: Menu-Driven Array Arithmetic

#include <stdio.h>

int main() {

int arr1[5] = {10, 20, 30, 40, 50};

int arr2[5] = {5, 4, 3, 2, 1};

int result[5];

int size = 5;

int i, choice;

do {

printf("\n--- Array Arithmetic Menu ---\n");

printf("1. Add Arrays\n");

printf("2. Subtract Arrays\n");

printf("3. Multiply Arrays\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: // Addition

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

result[i] = arr1[i] + arr2[i];

}

printf("Result of Addition: ");

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

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

}

printf("\n");

break;

case 2: // Subtraction

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

result[i] = arr1[i] - arr2[i];

}

printf("Result of Subtraction: ");

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

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

}

printf("\n");

break;

case 3: // Multiplication

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

result[i] = arr1[i] \* arr2[i];

}

printf("Result of Multiplication: ");

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

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

}

printf("\n");

break;

case 4:

printf("Exiting...\n");

break;

default:

printf("Invalid choice!\n");

}

} while (choice != 4);

return 0;

}

**Sample Output:**

**--- Array Arithmetic Menu ---**

**1. Add Arrays**

**2. Subtract Arrays**

**3. Multiply Arrays**

**4. Exit**

**Enter your choice: 1**

**Result of Addition: 15 24 33 42 51**

**--- Array Arithmetic Menu ---**

**1. Add Arrays**

**2. Subtract Arrays**

**3. Multiply Arrays**

**4. Exit**

**Enter your choice: 3**

**Result of Multiplication: 50 80 90 80 50**

**--- Array Arithmetic Menu ---**

**1. Add Arrays**

**2. Subtract Arrays**

**3. Multiply Arrays**

**4. Exit**

**Enter your choice: 4**

**Exiting...**

Question 5: Merge Two Sorted Arrays

#include <stdio.h>

int main() {

int arr1[5] = {1, 5, 10, 15, 20};

int size1 = 5;

int arr2[4] = {2, 3, 12, 18};

int size2 = 4;

int merged\_arr[9];

int i = 0, j = 0, k = 0; // Pointers for arr1, arr2, and merged\_arr

// Main loop to compare and merge

while (i < size1 && j < size2) {

if (arr1[i] < arr2[j]) {

merged\_arr[k++] = arr1[i++];

} else {

merged\_arr[k++] = arr2[j++];

}

}

// If there are leftover elements in arr1, copy them

while (i < size1) {

merged\_arr[k++] = arr1[i++];

}

// If there are leftover elements in arr2, copy them

while (j < size2) {

merged\_arr[k++] = arr2[j++];

}

printf("Merged Sorted Array: ");

for (i = 0; i < size1 + size2; i++) {

printf("%d ", merged\_arr[i]);

}

printf("\n");

return 0;

}

**Sample Output:**

**Merged Sorted Array: 1 2 3 5 10 12 15 18 20**

Question 6: Programs 1, 2, 3 using Functions & Call by Address

#include <stdio.h>

// --- Function for Question 1: Linear Search ---

// Returns 1 if found, 0 if not

int linearSearch(int \*arr, int size, int key) {

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

if (\*(arr + i) == key) { // \*(arr + i) is the same as arr[i]

return 1; // Found

}

}

return 0; // Not found

}

// --- Function for Question 2: Second Max/Min ---

// Uses pointers to "return" two values

void findSecondMaxMin(int \*arr, int size, int \*secondMin, int \*secondMax) {

// Simple bubble sort

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

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

if (\*(arr + j) > \*(arr + j + 1)) {

int temp = \*(arr + j);

\*(arr + j) = \*(arr + j + 1);

\*(arr + j + 1) = temp;

}

}

}

// Modify the values in main() using the pointers

\*secondMin = \*(arr + 1);

\*secondMax = \*(arr + size - 2);

}

// --- Functions for Question 3: Array Operations ---

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

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

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

}

printf("\n");

}

void insert(int \*arr, int \*size\_ptr, int pos, int val) {

for (int i = \*size\_ptr - 1; i >= pos; i--) {

\*(arr + i + 1) = \*(arr + i);

}

\*(arr + pos) = val;

(\*size\_ptr)++; // Increment the original size in main()

}

void deleteElement(int \*arr, int \*size\_ptr, int pos) {

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

\*(arr + i) = \*(arr + i + 1);

}

(\*size\_ptr)--; // Decrement the original size in main()

}

int main() {

printf("--- Q1: Linear Search using Function ---\n");

int arr1[] = {10, 20, 30, 40};

if (linearSearch(arr1, 4, 30)) {

printf("Number 30 was found.\n");

}

printf("\n--- Q2: Second Min/Max using Function ---\n");

int arr2[] = {5, 1, 9, 2, 8};

int sMin, sMax;

findSecondMaxMin(arr2, 5, &sMin, &sMax); // Pass addresses of sMin and sMax

printf("Second Min: %d, Second Max: %d\n", sMin, sMax);

printf("\n--- Q3: Array Ops using Functions ---\n");

int arr3[20] = {100, 200, 300, 400};

int size3 = 4;

printf("Original: ");

traverse(arr3, size3);

insert(arr3, &size3, 2, 250); // Pass address of size3

printf("After Insert: ");

traverse(arr3, size3);

deleteElement(arr3, &size3, 1); // Pass address of size3

printf("After Delete: ");

traverse(arr3, size3);

return 0;

}

**Sample Output:**

**--- Q1: Linear Search using Function ---**

**Number 30 was found.**

**--- Q2: Second Min/Max using Function ---**

**Second Min: 2, Second Max: 8**

**--- Q3: Array Ops using Functions ---**

**Original: 100 200 300 400**

**After Insert: 100 200 250 300 400**

**After Delete: 100 250 300 400**