Data Structures and algorithms

Submitted by:

Aryaveer Singh

102304064

3D12

Assignment 2

Q1: Write a program to check whether a given number is present in an array or not (Linear search).

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int size = sizeof(arr) / sizeof(arr[0]);
    int search_num, found = 0, i;

    printf("Enter the number to search for: ");
    scanf("%d", &search_num);

    for (i = 0; i < size; i++) {
        if (arr[i] == search_num) {
            found = 1;
            break;
        }
    }
```

```
    if (found) {
        printf("%d is present in the array at index %d.\n", search_num, i);
    } else {
        printf("%d is not present in the array.\n", search_num);
    }

    return 0;
}
```

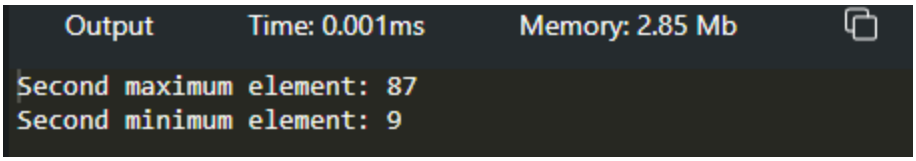
```
nter the number to search for: 0 is not present in the
   array.
```

Q2: Write a program to get second maximum and second minimum elements in an array.

```
#include <stdio.h>
#include <limits.h>

int main() {
    int arr[] = {12, 45, 1, 9, 87, 33, 22, 98};
    int n = sizeof(arr) / sizeof(arr[0]);
    int max1, max2, min1, min2;
    int i;

    if (n < 2) {
        printf("Array should have at least two elements.\n");
        return 1;
    }

     Initialize first and second max/min
```

```
if (arr[0] > arr[1]) {
    max1 = arr[0];
    max2 = arr[1];
    min1 = arr[1];
    min2 = arr[0];
} else {
    max1 = arr[1];
    max2 = arr[0];
    min1 = arr[0];
    min2 = arr[1];
}

for (i = 2; i < n; i++) {
     Check for max
    if (arr[i] > max1) {
        max2 = max1;
        max1 = arr[i];
    } else if (arr[i] > max2 && arr[i] < max1) {
        max2 = arr[i];
    }

     Check for min
    if (arr[i] < min1) {
        min2 = min1;
        min1 = arr[i];
    } else if (arr[i] < min2 && arr[i] > min1) {
        min2 = arr[i];
    }
}
```

```
    printf("Second maximum element: %d\n", max2);
    printf("Second minimum element: %d\n", min2);

    return 0;
}
```

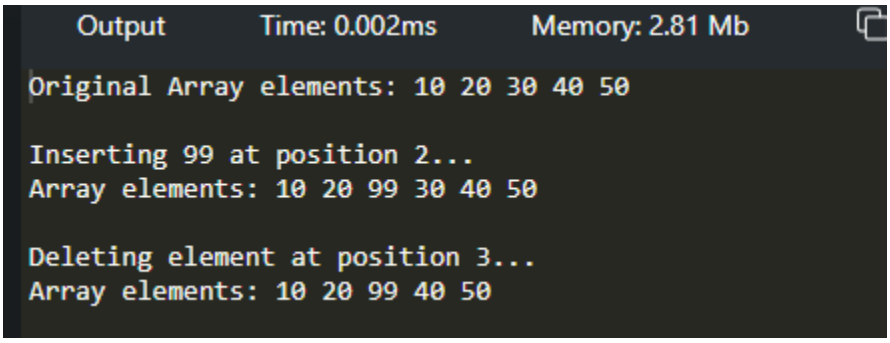


Q3: Write a program to perform insertion (any location), deletion (any location) and traversal in an array.

```
#include <stdio.h>
#define MAX_SIZE 100
 Function to traverse and print the array
void traverseArray(int arr[], int size) {
    printf("Array elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
 Function to insert an element at a given position
int insertElement(int arr[], int size, int element, int position) {
    if (size >= MAX_SIZE) {
        printf("Array is full. Insertion failed.\n");
        return size;
    }
    if (position < 0 || position > size) {
```

```
        printf("Invalid position. Insertion failed.\n");
        return size;
    }

    for (int i = size; i > position; i--) {
        arr[i] = arr[i - 1];
    }
    arr[position] = element;
    return size + 1;
}

 Function to delete an element at a given position
int deleteElement(int arr[], int size, int position) {
    if (position < 0 || position >= size) {
        printf("Invalid position. Deletion failed.\n");
        return size;
    }

    for (int i = position; i < size - 1; i++) {
        arr[i] = arr[i + 1];
    }
    return size - 1;
}

int main() {
    int arr[MAX_SIZE] = {10, 20, 30, 40, 50};
    int size = 5;

    printf("Original ");
```

```
    traverseArray(arr, size);

     Insertion
    int element_to_insert = 99, insert_pos = 2;
    printf("\nInserting %d at position %d...\n", element_to_insert, insert_pos);
    size = insertElement(arr, size, element_to_insert, insert_pos);
    traverseArray(arr, size);

     Deletion
    int delete_pos = 3;
    printf("\nDeleting element at position %d...\n", delete_pos);
    size = deleteElement(arr, size, delete_pos);
    traverseArray(arr, size);

    return 0;
}
```



Q4: Write a menu driven program to perform addition, multiplication and subtraction of 2 arrays.

```
#include <stdio.h>

void displayArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void addArrays(int arr1[], int arr2[], int result[], int size) {
    for (int i = 0; i < size; i++) {
        result[i] = arr1[i] + arr2[i];
    }
}

void subtractArrays(int arr1[], int arr2[], int result[], int size) {
    for (int i = 0; i < size; i++) {
        result[i] = arr1[i] - arr2[i];
    }
}

void multiplyArrays(int arr1[], int arr2[], int result[], int size) {
    for (int i = 0; i < size; i++) {
        result[i] = arr1[i] * arr2[i];
    }
}

int main() {
    int arr1[] = {1, 2, 3, 4, 5};
    int arr2[] = {5, 4, 3, 2, 1};
    int size = 5;
    int result[5];
    int choice;
```

```
do {
    printf("\nMenu:\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:
            addArrays(arr1, arr2, result, size);
            printf("Result of Addition: ");
            displayArray(result, size);
            break;
        case 2:
            subtractArrays(arr1, arr2, result, size);
            printf("Result of Subtraction: ");
            displayArray(result, size);
            break;
        case 3:
            multiplyArrays(arr1, arr2, result, size);
            printf("Result of Multiplication: ");
            displayArray(result, size);
            break;
        case 4:
            printf("Exiting program.\n");
            break;
```

```
        default:
            printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```

```
Menu:
1. Add arrays
2. Subtract arrays
3. Multiply arrays
4. Exit
Enter your choice: 2
Result of Subtraction: -4 -2 0 2 4

Menu:
1. Add arrays
2. Subtract arrays
3. Multiply arrays
4. Exit
Enter your choice: 4
Exiting program.
```

Q5: Write a program to perform sorting while merging (Merge two sorted arrays into one sorted array).

```
#include <stdio.h>

void mergeSortedArrays(int arr1[], int size1, int arr2[], int size2, int result[]) {
    int i = 0, j = 0, k = 0;

    Merge the arrays
    while (i < size1 && j < size2) {
        if (arr1[i] < arr2[j]) {
            result[k++] = arr1[i++];
        } else {
            result[k++] = arr2[j++];
```

```
        }
    }

    Copy remaining elements of arr1
    while (i < size1) {
        result[k++] = arr1[i++];
    }

    Copy remaining elements of arr2
    while (j < size2) {
        result[k++] = arr2[j++];
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr1[] = {1, 3, 5, 7, 9};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);

    int arr2[] = {2, 4, 6, 8, 10};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);

    int result_size = size1 + size2;
```

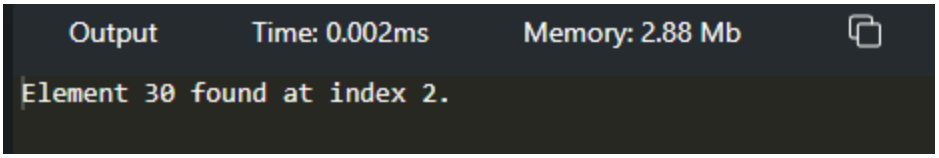
```
    int result[result_size];

    mergeSortedArrays(arr1, size1, arr2, size2, result);

    printf("Merged and sorted array: ");
    printArray(result, result_size);

    return 0;
}
```

Q6: Write the above programs (1, 2, and 3) using functions and call by address only.

Q1 (Revisited): Linear Search with Call by Address

```
#include <stdio.h>

int linearSearch(int *arr, int size, int *search_num, int *found_index) {
    for (int i = 0; i < size; i++) {
        if (*(arr + i) == *search_num) {
            *found_index = i;
            return 1;  Found
        }
    }
    return 0;  Not found
}

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int size = 5;
```

```
    int search_num = 30;
    int found_index = -1;

    if (linearSearch(arr, size, &search_num, &found_index)) {
        printf("Element %d found at index %d.\n", search_num, found_index);
    } else {
        printf("Element %d not found.\n", search_num);
    }

    return 0;
}
```

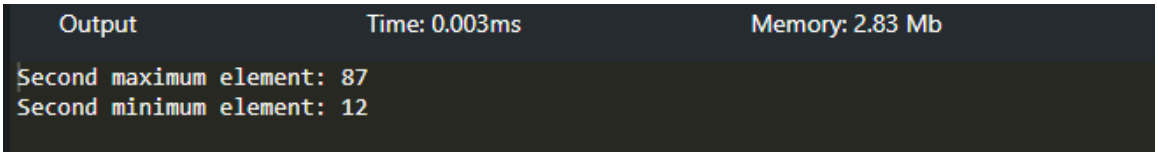


Q2 (Revisited): Second Max/Min with Call by Address

```
#include <stdio.h>

void findSecondMaxMin(int *arr, int size, int *second_max, int *second_min) {
    int max1, max2, min1, min2;
    int i;

    if (size < 2) {
        printf("Array should have at least two elements.\n");
        return;
    }

    if (arr[0] > arr[1]) {
```

```
        max1 = arr[0];
        max2 = arr[1];
        min1 = arr[1];
        min2 = arr[0];
    } else {
        max1 = arr[1];
        max2 = arr[0];
        min1 = arr[0];
        min2 = arr[1];
    }

    for (i = 2; i < size; i++) {
        if (arr[i] > max1) {
            max2 = max1;
            max1 = arr[i];
        } else if (arr[i] > max2 && arr[i] < max1) {
            max2 = arr[i];
        }

        if (arr[i] < min1) {
            min2 = min1;
            min1 = arr[i];
        } else if (arr[i] < min2 && arr[i] > min1) {
            min2 = arr[i];
        }
    }
    *second_max = max2;
    *second_min = min2;
}
```

```
int main() {
    int arr[] = {12, 45, 1, 9, 87, 33, 22, 98};
    int size = sizeof(arr) / sizeof(arr[0]);
    int second_max, second_min;

    findSecondMaxMin(arr, size, &second_max, &second_min);
    printf("Second maximum element: %d\n", second_max);
    printf("Second minimum element: %d\n", second_min);

    return 0;
}
```



Q3 (Revisited): Array operations with Call by Address

```
#include <stdio.h>
void traverse(int *arr, int size) {
    printf("Array elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", *(arr + i));
    }
    printf("\n");
}

int insert(int *arr, int *size, int element, int position) {
    if (position < 0 || position > *size) {
```

```
        printf("Invalid position for insertion.\n");
        return 0;
    }
    for (int i = *size; i > position; i--) {
        *(arr + i) = *(arr + i - 1);
    }
    *(arr + position) = element;
    (*size)++;
    return 1;
}

int delete(int *arr, int *size, int position) {
    if (position < 0 || position >= *size) {
        printf("Invalid position for deletion.\n");
        return 0;
    }
    for (int i = position; i < *size - 1; i++) {
        *(arr + i) = *(arr + i + 1);
    }
    (*size)--;
    return 1;
}

int main() {
    int arr[100] = {10, 20, 30, 40, 50};
    int size = 5;

    printf("Original ");
    traverse(arr, size);
```

```
    insert(arr, &size, 99, 2);
    printf("After insertion: ");
    traverse(arr, size);

    delete(arr, &size, 3);
    printf("After deletion: ");
    traverse(arr, size);

    return 0;
}
```