DSA ASSIGNMENT

Practical 1

Name: Madhav Jha
Roll no.: 48
Branch: CSE (AI & ML)

Q1. Write a program to swap numbers using call by value and call by reference and also print the difference in output.

a.Swap by value

```c
#include <stdio.h>

void swapByValue(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
    printf("\nSwapped in function:\na: %d; b: %d", a, b);
}

int main()
{
    int a = 10, b = 15;

    printf("Call by value\n");
    printf("Before swap:\na: %d; b: %d", a, b);
    swapByValue(a, b);
    printf("\nAfter swap:\na: %d; b: %d", a, b);

    return 0;
}
```

Output:

```
Call by value
Before swap:
a: 10; b: 15
Swapped in function:
a: 15; b: 10
After swap:
a: 10; b: 15
```

b.Swap by reference

```c
#include <stdio.h>

void swapByReference(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
    printf("\nSwapped in function:\na: %d; b: %d", *a, *b);
}

int main()
{
    int a = 10, b = 15;

    printf("\n\nCall by Reference\n");
    printf("Before swap:\na: %d; b: %d", a, b);
    swapByReference(&a, &b);
    printf("\nAfter swap:\na: %d; b: %d", a, b);

    return 0;
}
```

Output:

```
Call by Reference
Before swap:
a: 10; b: 15
Swapped in function:
a: 15; b: 10
After swap:
a: 15; b: 10
```

Q2. Write a program to return multiple values through c program by following ways

a.Using pointers

```c
#include <stdio.h>

void initialize(int* a, int* b, char* c)
{
    *a = 10;
    *b = 20;
    *c = 'A';
}

int main(void)
{
    int a, b;
    char c;

    initialize(&a, &b, &c);
    printf("a = %d, b = %d, c = %c", a, b, c);

    return 0;
}
```

Output:

```
a = 10, b = 20, c = A
```

b.Using structure

```c
#include <stdio.h>

struct Tuple {
    int a, b;
    char c;
};

struct Tuple initialize()
{
    struct Tuple tuple = { 10, 20, 'A' };

    return tuple;
}

int main(void)
{
    int a, b;
    char c;

    struct Tuple tuple = initialize();

    a = tuple.a;
    b = tuple.b;
    c = tuple.c;

    printf("a = %d, b = %d, c = %c", a, b, c);

    return 0;
}
```

Output:

```
a = 10, b = 20, c = A
```

c.Using Array

```c
#include <stdio.h>
#include <stdlib.h>

int* initialize()
{
    int* temp = (int*)malloc(sizeof(int) * 3);

    *temp = 10;
    *(temp + 1) = 20;
    *(temp + 2) = 30;

    return temp;
}

int main(void)
{
    int a, b, c;

    int* arr = initialize();

    a = arr[0];
    b = arr[1];
    c = arr[2];

    printf("a = %d, b = %d, c = %d", a, b, c);

    return 0;
}
```

Output:

```
a = 10, b = 20, c = 30
```

Q3.Search element in array

a.Linear search

```c
#include <stdio.h>

void linearSearch(int arr[], int* n, int* num, int* index) {
    for (int i = 0;i < *n;i++) {
        if (arr[i] == *num) {
            *index = i;
            break;
        }
    }
}

int main() {
    int n = 2;
    printf("\nEnter number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter array elements: ");
    for (int i = 0;i < n;i++) {
        scanf("%d", &arr[i]);
    }

    int num;
    printf("Enter number to search: ");
    scanf("%d", &num);

    int index = -1;
    linearSearch(arr, &n, &num, &index);

    if (index == -1) {
        printf("Number not found!!!");
    }
    else {
```

```
        printf("Number found at index: %d", index);
    }
    printf("\n\n");
    return 0;
}
```

Output:

```
Enter number of elements: 5
Enter array elements: 6 5 4 3 2
Enter number to search: 4
Number found at index: 2
```

b.Binary search

```c
#include <stdio.h>

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

int* bubbleSort(int arr[], int n) {
    for (int i = 0;i < n;i++) {
        for (int j = 0;j < n;j++) {
            if (arr[i] < arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    return arr;
}

void binarySearch(int arr[], int* n, int* num, int* index) {
    int l = 0, r = *n - 1, m;
    int count = 0;
    while (l <= r && count < 100) {
        m = (l + r) / 2;
        if (arr[m] == *num) {
            *index = m;
            break;
        }
        else if (arr[m] < *num) {
            l = m + 1;
        }
        else {
            r = m;
        }
        count++;
```

```
    }
}

int main() {
    int n = 2;
    printf("\nEnter number of elements: ");
    scanf("%d", &n);

    int* arr;
    printf("Enter array elements: ");
    for (int i = 0;i < n;i++) {
        scanf("%d", &arr[i]);
    }

    printf("After sorting: ");
    arr = bubbleSort(arr, n);
    printArray(arr, n);

    int num;
    printf("\nEnter number to search: ");
    scanf("%d", &num);

    int index = -1;
    binarySearch(arr, &n, &num, &index);

    if (index == -1) {
        printf("Number not found!!!");
    }
    else {
        printf("Number found at index: %d", index);
    }
    printf("\n\n");
    return 0;
}
```

Output:

```
Enter number of elements: 10
Enter array elements: 11 10 9 8 7 6 5 5 4 3
After sorting: [ 3 4 5 5 6 7 8 9 10 11 ]
Enter number to search: 6
Number found at index: 4
```