My C language files - 01_Linear_Search.c

```c
// C code to linearly search x in arr[]. If x
// is present then return its location, otherwise
// return -1

#include <stdio.h>

int search(int arr[], int N, int x)
{
    int i;
    for (i = 0; i < N; i++)
        if (arr[i] == x)
            return i;
    return -1;
}

// Driver code
int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int N = sizeof(arr) / sizeof(arr[0]);

    // Function call
    int result = search(arr, N, x);
    (result == -1)
        ? printf("Element is not present in array")
        : printf("Element is present at index %d", result);
    return 0;
}

```

My C language files - 02_binary_search.c

```c
#include<stdio.h>


int binarySearch(int arr[], int size, int element){
    int low, mid, high;
    low = 0;
    high = size-1;
    // Keep searching until low <= high
    while(low<=high){
        mid = (low + high)/2;
        if(arr[mid] == element){
            return mid;
        }
        if(arr[mid]<element){
            low = mid+1;
        }
        else{
            high = mid -1;
        }
    }
    return -1;

}

int main(){

    // Sorted array for binary search
    int arr[] = {1,3,5,56,64,73,123,225,444};
    int size = sizeof(arr)/sizeof(int);
    int element = 444;
    int searchIndex = binarySearch(arr, size, element);
    printf("The element %d was found at index %d \n", element, searchIndex);
    return 0;
}

```

My C language files - 01_Bubble_sort.c

```c
// C program for implementation of Bubble sort
#include <stdio.h>

void swap(int* xp, int* yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(&arr[j], &arr[j + 1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program to test above functions
int main()
{
    int arr[] = { 5, 1, 4, 2, 8 };
    int n = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}

```

My C language files - 02_insertion_sort.c

```c
#include<stdio.h>

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

void insertionSort(int *A, int n){
    int key, j;
    // Loop for passes
    for (int i = 1; i <= n-1; i++)
    {
        key = A[i];
        j = i-1;
        // Loop for each pass
        while(j>=0 && A[j] > key){
            A[j+1] = A[j];
            j--;
        }
        A[j+1] = key;
    }
}

int main(){
   // -1   0    1   2   3   4   5
   //      12,| 54, 65, 07, 23, 09 --> i=1, key=54, j=0
   //      12,| 54, 65, 07, 23, 09 --> 1st pass done (i=1)!

   //      12, 54,| 65, 07, 23, 09 --> i=2, key=65, j=1
   //      12, 54,| 65, 07, 23, 09 --> 2nd pass done (i=2)!

   //      12, 54, 65,| 07, 23, 09 --> i=3, key=7, j=2
   //      12, 54, 65,| 65, 23, 09 --> i=3, key=7, j=1
   //      12, 54, 54,| 65, 23, 09 --> i=3, key=7, j=0
   //      12, 12, 54,| 65, 23, 09 --> i=3, key=7, j=-1
   //      07, 12, 54,| 65, 23, 09 --> i=3, key=7, j=-1--> 3rd pass done (i=3)!

   // Fast forwarding and 4th and 5th pass will give:
   //      07, 12, 54, 65,| 23, 09 --> i=4, key=23, j=3
   //      07, 12, 23, 54,| 65, 09 --> After the 4th pass

   //      07, 12, 23, 54, 65,| 09 --> i=5, key=09, j=4
   //      07, 09, 12, 23, 54, 65| --> After the 5th pass



    int A[] = {12, 54, 65, 7, 23, 9};
    int n = 6;
    printArray(A, n);
    insertionSort(A, n);
    printArray(A, n);
    return 0;
}

```

My C language files - 03_selection_sort.c

```c
// C program for implementation of selection sort
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
        if(min_idx != i)
            swap(&arr[min_idx], &arr[i]);
    }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program to test above functions
int main()
{
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}

```

My C language files - Test_Code.c

```c
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(&arr[j], &arr[j + 1]);
}


void insertionSort(int *A, int n){
    int key, j;
    // Loop for passes
    for (int i = 1; i <= n-1; i++)
    {
        key = A[i];
        j = i-1;
        // Loop for each pass
        while(j>=0 && A[j] > key){
            A[j+1] = A[j];
            j--;
        }
        A[j+1] = key;
    }
}


void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    // One by one move boundary of unsorted subarray
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;

        // Swap the found minimum element with the first element
        if(min_idx != i)
            swap(&arr[min_idx], &arr[i]);
    }
}

```