

```
1 // C code to linearly search x in arr[]. If x
2 // is present then return its location, otherwise
3 // return -1
4
5 #include <stdio.h>
6
7 int search(int arr[], int N, int x)
8 {
9     int i;
10    for (i = 0; i < N; i++)
11        if (arr[i] == x)
12            return i;
13    return -1;
14 }
15
16 // Driver code
17 int main(void)
18 {
19     int arr[] = { 2, 3, 4, 10, 40 };
20     int x = 10;
21     int N = sizeof(arr) / sizeof(arr[0]);
22
23     // Function call
24     int result = search(arr, N, x);
25     (result == -1)
26         ? printf("Element is not present in array")
27         : printf("Element is present at index %d", result);
28     return 0;
29 }
30
```

```
1  #include<stdio.h>
2
3
4  int binarySearch(int arr[], int size, int element){
5      int low, mid, high;
6      low = 0;
7      high = size-1;
8      // Keep searching until low <= high
9      while(low<=high){
10         mid = (low + high)/2;
11         if(arr[mid] == element){
12             return mid;
13         }
14         if(arr[mid]<element){
15             low = mid+1;
16         }
17         else{
18             high = mid -1;
19         }
20     }
21     return -1;
22 }
23
24
25 int main(){
26
27     // Sorted array for binary search
28     int arr[] = {1,3,5,56,64,73,123,225,444};
29     int size = sizeof(arr)/sizeof(int);
30     int element = 444;
31     int searchIndex = binarySearch(arr, size, element);
32     printf("The element %d was found at index %d \n", element, searchIndex);
33     return 0;
34 }
35
```

```
1 // C program for implementation of Bubble sort
2 #include <stdio.h>
3
4 void swap(int* xp, int* yp)
5 {
6     int temp = *xp;
7     *xp = *yp;
8     *yp = temp;
9 }
10
11 // A function to implement bubble sort
12 void bubbleSort(int arr[], int n)
13 {
14     int i, j;
15     for (i = 0; i < n - 1; i++)
16
17         // Last i elements are already in place
18         for (j = 0; j < n - i - 1; j++)
19             if (arr[j] > arr[j + 1])
20                 swap(&arr[j], &arr[j + 1]);
21 }
22
23 /* Function to print an array */
24 void printArray(int arr[], int size)
25 {
26     int i;
27     for (i = 0; i < size; i++)
28         printf("%d ", arr[i]);
29     printf("\n");
30 }
31
32 // Driver program to test above functions
33 int main()
34 {
35     int arr[] = { 5, 1, 4, 2, 8 };
36     int n = sizeof(arr) / sizeof(arr[0]);
37     bubbleSort(arr, n);
38     printf("Sorted array: \n");
39     printArray(arr, n);
40     return 0;
41 }
42
```

```

1  #include<stdio.h>
2
3  void printArray(int* A, int n){
4      for (int i = 0; i < n; i++)
5      {
6          printf("%d ", A[i]);
7      }
8      printf("\n");
9  }
10
11 void insertionSort(int *A, int n){
12     int key, j;
13     // Loop for passes
14     for (int i = 1; i <= n-1; i++)
15     {
16         key = A[i];
17         j = i-1;
18         // Loop for each pass
19         while(j>=0 && A[j] > key){
20             A[j+1] = A[j];
21             j--;
22         }
23         A[j+1] = key;
24     }
25 }
26
27 int main(){
28     // -1   0    1    2    3    4    5
29     //      12, | 54, 65, 07, 23, 09 --> i=1, key=54, j=0
30     //      12, | 54, 65, 07, 23, 09 --> 1st pass done (i=1)!
31
32     //      12, 54, | 65, 07, 23, 09 --> i=2, key=65, j=1
33     //      12, 54, | 65, 07, 23, 09 --> 2nd pass done (i=2)!
34
35     //      12, 54, 65, | 07, 23, 09 --> i=3, key=7, j=2
36     //      12, 54, 65, | 65, 23, 09 --> i=3, key=7, j=1
37     //      12, 54, 54, | 65, 23, 09 --> i=3, key=7, j=0
38     //      12, 12, 54, | 65, 23, 09 --> i=3, key=7, j=-1
39     //      07, 12, 54, | 65, 23, 09 --> i=3, key=7, j=-1--> 3rd pass done (i=3)!
40
41     // Fast forwarding and 4th and 5th pass will give:
42     //      07, 12, 54, 65, | 23, 09 --> i=4, key=23, j=3
43     //      07, 12, 23, 54, | 65, 09 --> After the 4th pass
44
45     //      07, 12, 23, 54, 65, | 09 --> i=5, key=09, j=4
46     //      07, 09, 12, 23, 54, 65 | --> After the 5th pass
47
48
49
50     int A[] = {12, 54, 65, 7, 23, 9};
51     int n = 6;
52     printArray(A, n);
53     insertionSort(A, n);
54     printArray(A, n);
55     return 0;
56 }
57

```



```
1 // C program for implementation of selection sort
2 #include <stdio.h>
3
4 void swap(int *xp, int *yp)
5 {
6     int temp = *xp;
7     *xp = *yp;
8     *yp = temp;
9 }
10
11 void selectionSort(int arr[], int n)
12 {
13     int i, j, min_idx;
14
15     // One by one move boundary of unsorted subarray
16     for (i = 0; i < n-1; i++)
17     {
18         // Find the minimum element in unsorted array
19         min_idx = i;
20         for (j = i+1; j < n; j++)
21             if (arr[j] < arr[min_idx])
22                 min_idx = j;
23
24         // Swap the found minimum element with the first element
25         if(min_idx != i)
26             swap(&arr[min_idx], &arr[i]);
27     }
28 }
29
30 /* Function to print an array */
31 void printArray(int arr[], int size)
32 {
33     int i;
34     for (i=0; i < size; i++)
35         printf("%d ", arr[i]);
36     printf("\n");
37 }
38
39 // Driver program to test above functions
40 int main()
41 {
42     int arr[] = {64, 25, 12, 22, 11};
43     int n = sizeof(arr)/sizeof(arr[0]);
44     selectionSort(arr, n);
45     printf("Sorted array: \n");
46     printArray(arr, n);
47     return 0;
48 }
49
```

```
1 void bubbleSort(int arr[], int n)
2 {
3     int i, j;
4     for (i = 0; i < n - 1; i++)
5
6         // Last i elements are already in place
7         for (j = 0; j < n - i - 1; j++)
8             if (arr[j] > arr[j + 1])
9                 swap(&arr[j], &arr[j + 1]);
10 }
11
12
13 void insertionSort(int *A, int n){
14     int key, j;
15     // Loop for passes
16     for (int i = 1; i <= n-1; i++)
17     {
18         key = A[i];
19         j = i-1;
20         // Loop for each pass
21         while(j>=0 && A[j] > key){
22             A[j+1] = A[j];
23             j--;
24         }
25         A[j+1] = key;
26     }
27 }
28
29
30 void selectionSort(int arr[], int n)
31 {
32     int i, j, min_idx;
33
34     // One by one move boundary of unsorted subarray
35     for (i = 0; i < n-1; i++)
36     {
37         // Find the minimum element in unsorted array
38         min_idx = i;
39         for (j = i+1; j < n; j++)
40             if (arr[j] < arr[min_idx])
41                 min_idx = j;
42
43         // Swap the found minimum element with the first element
44         if(min_idx != i)
45             swap(&arr[min_idx], &arr[i]);
46     }
47 }
48
```