```
void swap(int *a, int *b)
{
int temp = *a;
*a = *b;
*b = temp;
}
```

```
// bubble sort function
void bubbleSort(int array[], int n)
{
int i, j;
for (i = 0; i < n-1; i++)
for (j = 0; j < n-i-1; j++)
     if (array[j] > array[j+1])
     swap(&array[j], &array[j+1]);
}
```

```
// Insertion Sort Function
void insertionSort (int array[], int n)
{
int i, element, j;
for (i = 1; i < n; i++)
    {
      element = array[i];
      j = i - 1;
      while (j >= 0 && array[j] > element)
     {
array[j + 1] = array[j];
j = j - 1;
}
array[j + 1] = element;
}
```

```
// Selection Sort
void selectionSort(int array[], int n)
{
int i, j, min_element;
for (i = 0; i < n-1; i++)
{
min_element = i;
for (j = i+1; j < n; j++)
if (array[j] < array[min_element])
min_element = j;
swap(&array[min_element], &array[i]);
}
}

// QuickSort Function
#include <stdio.h>

void qs(int A[], int low, int high) {
    if (low < high) {
        int pivot = low;
        int i = low;
        int j = high;
        while (i < j) {
            while (A[i] <= A[pivot] && i < high) {
                i++;
            }
            while (A[j] > A[pivot]) {
                j--;
            }
            if (i < j) {
                int temp = A[i];
                A[i] = A[j];
                A[j] = temp;
            }
        }
        int temp = A[pivot];
        A[pivot] = A[j];
        A[j] = temp;

        qs(A, low, j - 1);
        qs(A, j + 1, high);
    }
}
```

```
int main() {
    int A[6] = {1, 6, 4, 8, 2, 5};
    int n = 6;
    qs(A, 0, n - 1);
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);

    return 0;
}
```

```
// Merge sort in C

#include <stdio.h>


void merge(int arr[], int p, int q, int r) {


  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];

  int i, j, k;
  i = 0;
  j = 0;
  k = p;

  while (i < n1 && j < n2) {
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
```

```
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }

  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}


void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);
    merge(arr, l, m, r);
  }
}

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

// Driver program
int main() {
  int arr[] = {6, 5, 12, 10, 9, 1};
```

```
  int size = sizeof(arr) / sizeof(arr[0]);

  mergeSort(arr, 0, size - 1);

  printf("Sorted array: \n");
  printArray(arr, size);
}
```

//Binary Search

```
#include <stdio.h>

int binarySearch(int arr[], int l, int r, int x)
{
        if (r >= l) {
        int mid = l + (r - l) / 2;
                if (arr[mid] == x)
                        return mid;
                if (arr[mid] > x)
                        return binarySearch(arr, l, mid - 1, x);
                else
                return binarySearch(arr, mid + 1, r, x);
        }

        return -1;
}

// Driver code
int main()
{
        int arr[] = { 2, 3, 4, 10, 40 };
        int n = sizeof(arr) / sizeof(arr[0]);
        int x = 10;
        int result = binarySearch(arr, 0, n - 1, x);
        (result == -1)
                ? printf("Element is not present in array")
                : printf("Element is present at index %d", result);
        return 0;
}
```