1-

```cpp
#include <algorithm>
#include <iostream>
using namespace std;

// Function to return k'th smallest element in a given array
int kthSmallest(int arr[], int n, int k)
{
    // Sort the given array
    sort(arr, arr + n);

    // Return k'th element in the sorted array
    return arr[k - 1];
}

// Driver program to test above methods
int main()
{
    int arr[] = { 12, 3, 5, 7, 19 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout<<"enter the Kth value\n";int k;
    cin>>k;
    cout << "Kth smallest element is " << kthSmallest(arr, n, k);
    return 0;
}
```

enter the Kth value

3

K'th smallest element is 7

2-

```cpp
#include <iostream>
using namespace std;
int main()
{
    int facto(int);
    int fact, val;
    cout << "Enter any number: ";
    cin >> val;
    fact = facto(val);
    cout << "Factorial of a number is: " << fact << endl;
    return 0;
}
int facto(int n)
{
```

```
    if (n < 0)
        return (-1); /*Wrong value*/
    if (n == 0)
        return (1); /*Terminating condition*/
    else
    {
        return (n * facto(n - 1));
    }
}
```

Enter any number: 5

Factorial of a number is: 120

3-

```
#include <iostream>
using namespace std;

// function to swap elements
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// function to print the array
void printArray(int array[], int size) {
  int i;
  for (i = 0; i < size; i++)
    cout << array[i] << " ";
  cout << endl;
}

// function to rearrange array (find the partition point)
int partition(int array[], int low, int high) {

  // select the rightmost element as pivot
  int pivot = array[high];

  // pointer for greater element
  int i = (low - 1);

  // traverse each element of the array
  // compare them with the pivot
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {
```

```
      // if element smaller than pivot is found
      // swap it with the greater element pointed by i
      i++;

      // swap element at i with element at j
      swap(&array[i], &array[j]);
    }
  }

  // swap pivot with the greater element at i
  swap(&array[i + 1], &array[high]);

  // return the partition point
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {

    // find the pivot element such that
    // elements smaller than pivot are on left of pivot
    // elements greater than pivot are on righ of pivot
    int pi = partition(array, low, high);

    // recursive call on the left of pivot
    quickSort(array, low, pi - 1);

    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
  }
}

// Driver code
int main() {
  int data[] = {8, 7, 6, 1, 0, 9, 2};
  int n = sizeof(data) / sizeof(data[0]);

  cout << "Unsorted Array: \n";
  printArray(data, n);

  // perform quicksort on data
  quickSort(data, 0, n - 1);

  cout << "Sorted array in ascending order: \n";
  printArray(data, n);
}
```

Unsorted Array:

8 7 6 1 0 9 2

Sorted array in ascending order:

0 1 2 6 7 8 9