: Bubble sort algorithm

```
// C++ program for implementation of Bubble sort
#include <iostream>
using namespace std;

// 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++)
      cout << arr[i] << " ";
 cout << endl;
}
// Driver code
int main()
{
 int n, i;
 cout<<"\nEnter the number of data element to be sorted: ";
    cin>>n;
 int arr[n];
 for(i = 0; i < n; i++)
 {
      cout<<"Enter element "<<i+1<<": ";
      cin>>arr[i];
 }

 bubbleSort(arr, n);
 cout << "Sorted array: \n";
 printArray(arr, n);
    return 0;

 }
```

: Selection sort algorithm

```
// C++ program for implementation of selection sort
#include <iostream>
using namespace std;

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
      //swap(&arr[min_idx], &arr[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++)
      cout << arr[i] << " ";
 cout << endl;
}

// Driver program to test above functions
int main()
{
 int n, i;
 cout<<"\nEnter the number of data element to be sorted: ";
    cin>>n;
 int arr[n];
 for(i = 0; i < n; i++)
 {
      cout<<"Enter element "<<i+1<<": ";
      cin>>arr[i];
```

```
}

selectionSort(arr, n);
cout << "Sorted array: \n";
printArray(arr, n);
return 0;
}
```

: Insertion sort algorithm

```
// C++ program for insertion sort
#include <iostream>
using namespace std;

/* Function to sort an array using insertion sort*/
void insertionSort(int arr[], int n)
{
 int i, key, j;
 for (i = 1; i < n; i++)
 {
      key = arr[i];
      j = i - 1;

      /* Move elements of arr[0..i-1], that are
      greater than key, to one position ahead
      of their current position */
      while (j >= 0 && arr[j] > key)
      {
           arr[j + 1] = arr[j];
           j = j - 1;
      }
      arr[j + 1] = key;
 }
}

// A utility function to print an array of size n
void printArray(int arr[], int n)
{
 int i;
 for (i = 0; i < n; i++)
      cout << arr[i] << " ";
 cout << endl;
}

/* Driver code */
int main()
```

```
{

int n, i;
cout<<"\nEnter the number of data element to be sorted: ";
    cin>>n;
int arr[n];
for(i = 0; i < n; i++)
{
     cout<<"Enter element "<<i+1<<": ";
     cin>>arr[i];
}
    cout<<"\nSorted Data\n ";
insertionSort(arr, n);
printArray(arr, n);

return 0;
}
```

: Merge sort algorithm

```
// C++ program for implementation of merge sort
#include <iostream>
using namespace std;

// A function to merge the two half into a sorted data.
void Merge(int *a, int low, int high, int mid)
{
 // We have low to mid and mid+1 to high already sorted.
 int i, j, k, temp[high-low+1];
 i = low;
 k = 0;
 j = mid + 1;
 // Merge the two parts into temp[].
 while (i <= mid && j <= high)
 {
      if (a[i] < a[j])
      {
           temp[k] = a[i];
           k++;
           i++;
      }
      else
      {
           temp[k] = a[j];
```

```
            k++;
            j++;
      }
 }
 // Insert all the remaining values from i to mid into temp[].
 while (i <= mid)
 {
      temp[k] = a[i];
      k++;
      i++;
 }
 // Insert all the remaining values from j to high into
temp[].
 while (j <= high)
 {
      temp[k] = a[j];
      k++;
      j++;
 }
 // Assign sorted data stored in temp[] to a[].
 for (i = low; i <= high; i++)
 {
      a[i] = temp[i-low];
 }
}
// A function to split array into two parts.
void MergeSort(int *a, int low, int high)
{
 int mid;
 if (low < high)
 {
      mid=(low+high)/2;
      // Split the data into two half.
      MergeSort(a, low, mid);
      MergeSort(a, mid+1, high);

      // Merge them to get sorted output.
      Merge(a, low, high, mid);
 }
}
int main()
{
 int n, i;
 cout<<"\nEnter the number of data element to be sorted: ";
    cin>>n;
 int arr[n];
 for(i = 0; i < n; i++)
 {
```

```
      cout<<"Enter element "<<i+1<<": ";
      cin>>arr[i];
 }

 MergeSort(arr, 0, n-1);

 // Printing the sorted data.
 cout<<"\nSorted Data\n ";
 for (i = 0; i < n; i++)
         cout<<arr[i]<<" ";

 return 0;
}
```