**Q1**

**Bubble sort**

**//very basic type of bubble sort algorithm.**

**#include<iostream>**

**using namespace std;**

**void bubbleSort(int arr[],int n)**

**{**

**for(int i=1;i<n;i++)**

**{**

**for(int j=0;j<n-i;j++)**

**{**

**if(arr[j]>arr[j+1])**

**{**

**int temp=arr[j];**

**arr[j]=arr[j+1];**

**arr[j+1]=temp;**

**}**

**}**

**}**

**}**

**int main()**

**{**

**int a[] = {-2,45,0,11,-9};**

**int n = 5;// a way to find the size of the array.**

**cout<<n<<endl;**

**for(int i=0; i<n;i++)**

**{**

**cout<<a[i]<<" ";**

**}**

**cout<<endl<<"The sorted order of the array is "<<endl;**

**bubbleSort(a,n);**

**for(int i=0; i<n;i++)**

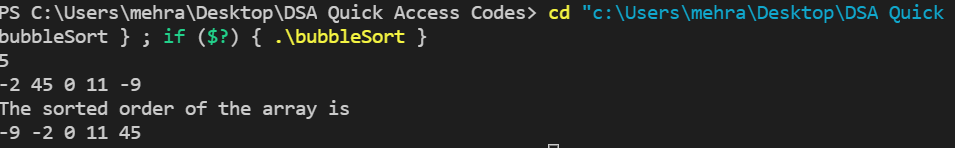
**{**

**cout<<a[i]<<" ";**

**}**

**return 0;**

**}**

****

**Selection Sort**

**#include<iostream>**

**using namespace std;**

**void selectionSort(int arr[],int n)**

**{**

**for(int i=0;i<n-1;i++)**

**{**

**for(int j=i;j<n;j++)**

**{**

**if(arr[j]<arr[i])**

**{**

**int temp=arr[j];**

**arr[j]=arr[i];**

**arr[i]=temp;**

**}**

**}**

**}**

**}**

**int main()**

**{**

**//find minimum element in the unsorted array and swap with the element at the beginning**

**int arr[6]={12,45,23,51,19,8};**

**selectionSort(arr,6);**

**for(int i =0;i<6;i++)**

**{**

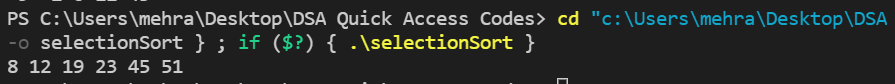
**cout<<arr[i]<<" ";**

**}**

**cout<<endl;**

**return 0;**

**}**

****

**Insertion Sort using arrays and linked list**

**#include<iostream>**

**using namespace std;**

**void selectionSort(int arr[],int n)**

**{**

**for(int i =0;i<n;i++)**

**{**

**int current=arr[i];**

**int j =i-1;**

**while(arr[i]>current && j>=0)**

**{**

**arr[j+1]=arr[j];**

**j--;**

**}**

**arr[j+1]=current;**

**}**

**}**

**int main()**

**{**

**//insert an element from unsorted array to its correct position in the sorted array**

**//correct position means that the elements before it must be lesser in value and the elements after it must be greater in value**

**int arr[]={12,45,23,51,19,8};**

**int n =6;**

**selectionSort(arr,n);**

**for(int i=0;i<n;i++)**

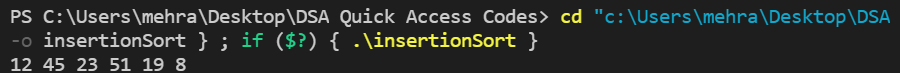
**{**

**cout<<arr[i]<<" ";**

**}**

**return 0;**

**}**

****

**Quick Sort**

**#include<iostream>**

**using namespace std;**

**int partition(int input[],int si,int ei)**

**{**

**int count=0;**

**for(int i =si+1;i<=ei;i++)**

**{**

**if(input[si]>=input[i]) //we first have to count how many numbers are less than the starting number.**

**count++;**

**}**

**int partitionIndex = si+count; //we have to add this as we are doing recursion.**

**//hence the partition is in place**

**int tempswap = input[si];**

**input[si] = input[partitionIndex];**

**input[partitionIndex]=tempswap;**

**//now we have to put numbers less than partition element before and the ones greater than it, after it.**

**int i=si, j =ei;**

**while(i!=partitionIndex&&j!=partitionIndex)**

**{**

**if(input[i]<=input[partitionIndex]) //if the element is less than, then move the pointer**

**i++;**

**else if(input[j]>=input[partitionIndex])**

**j++;**

**else**

**{**

**tempswap = input[i];**

**input[i]=input[j];**

**input[j]=tempswap;**

**}**

**}**

**return partitionIndex;**

**}**

**void qs(int input[], int si, int ei)**

**{**

**if(si>=ei)**

**return;**

**int partition\_index = partition(input,si,ei);**

**qs(input,si,partition\_index-1);**

**qs(input,partition\_index+1,ei);**

**}**

**void quickSort(int arr[],int size )**

**{**

**qs(input,0,size-1);**

**}**

**int main()**

**{**

**cout<<"Enter the number of terms in the dyanamic array\n";**

**cin>>n;**

**int \*input = new int[n];**

**for(int i =0;i<n;i++)**

**{**

**cin>>input[i];**

**}**

**quickSort(input,n);**

**for(int i =0;i<n;i++)**

**{**

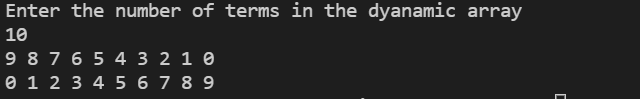
**cout<<input[i]<<" ";**

**}**

**delete [] input;**

**return 0;**

**}**

****

**MergeSort**

**#include<iostream>**

**using namespace std;**

**void merge(int arr[],int l,int mid,int r)**

**{**

**//make two temporary arrays to sort the 2 arrays and then merge then into the original arrays.**

**int n1 = mid-l+1;//as middle term is included**

**int n2 = r-mid;**

**int a[n1],b[n2];**

**//fillinng these 2 arrays**

**for(int i = 0;i<n1;i++)**

**{**

**a[i]=arr[l+i];**

**}**

**for(int i = 0;i<n2;i++)**

**{**

**b[i]=arr[mid+1+i];**

**}**

**int i=0,j=0,k=l;// k traverses the main array.**

**//comparing and filling the arrays**

**while(i<n1 && j<n2)**

**{**

**if(a[i]<b[j])**

**{**

**arr[k]=a[i];**

**i++; k++;**

**}**

**else**

**{**

**arr[k]=b[j];**

**j++;k++;**

**}**

**}**

**//there also might be cases where, the firfst array might finish before the second term.**

**while(i<n1)**

**{**

**arr[k]=a[i];**

**k++;i++;**

**}**

**while(j<n2)**

**{**

**arr[k]=b[j];**

**k++;j++;**

**}**

**}**

**void mergeSort(int arr[],int l,int r)**

**{**

**//l is the first index and r is the last index**

**if(l<r)**

**{**

**int mid = (l+r)/2;**

**mergeSort(arr,l,mid);**

**mergeSort(arr,mid+1,r);**

**merge(arr,l,mid,r); //merging the 2 sorted arrays**

**}**

**//when l=0, r=0, the returns its call. but if we didnt put this below return, our code would still return.**

**else**

**return;**

**}**

**int main()**

**{**

**arr[]={5,4,3,2,1};**

**for(int i =0;i<5;i++)**

**{**

**cout<<arr[i]<<" ";**

**}**

**cout<<endl;**

**mergeSort(arr,0,4);**

**for(int i=0;i<5;i++)**

**{**

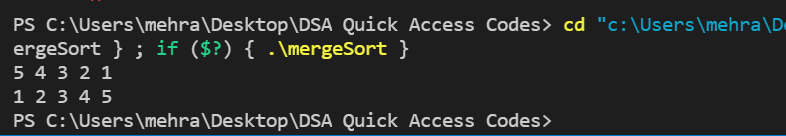
**cout<<arr[i]<<" ";**

**}**

**cout<<endl;**

**return 0;**

**}**

****

**Shell Sort**

**using namespace std;**

**#include <iostream>**

**void shellSort(int array[], int n) {**

**// Rearrange elements at each n/2, n/4, n/8, ... intervals**

**for (int interval = n / 2; interval > 0; interval /= 2) {**

**for (int i = interval; i < n; i += 1) {**

**int temp = array[i];**

**int j;**

**for (j = i; j >= interval && array[j - interval] > temp; j -= interval) {**

**array[j] = array[j - interval];**

**}**

**array[j] = temp;**

**}**

**}**

**}**

**void printArray(int array[], int size) {**

**int i;**

**for (i = 0; i < size; i++)**

**cout << array[i] << " ";**

**cout << endl;**

**}**

**int main() {**

**int data[] = {9, 8, 3, 7, 5, 6, 4, 1};**

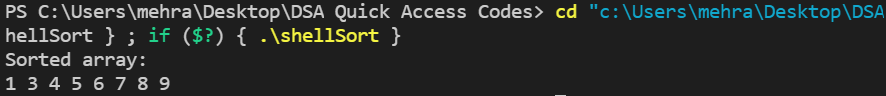
**int size = sizeof(data) / sizeof(data[0]);**

**shellSort(data, size);**

**cout << "Sorted array: \n";**

**printArray(data, size);**

**}**

****

**Count Sort**

**#include <iostream>**

**using namespace std;**

**void countSort(int array[], int size) {**

**int output[10];**

**int count[10];**

**int max = array[0];**

**for (int i = 1; i < size; i++) {**

**if (array[i] > max)**

**max = array[i];**

**}**

**for (int i = 0; i <= max; ++i) {**

**count[i] = 0;**

**}**

**for (int i = 0; i < size; i++) {**

**count[array[i]]++;**

**}**

**for (int i = 1; i <= max; i++) {**

**count[i] += count[i - 1];**

**}**

**for (int i = size - 1; i >= 0; i--) {**

**output[count[array[i]] - 1] = array[i];**

**count[array[i]]--;**

**}**

**for (int i = 0; i < size; i++) {**

**array[i] = output[i];**

**}**

**}**

**void printArray(int array[], int size) {**

**for (int i = 0; i < size; i++)**

**cout << array[i] << " ";**

**cout << endl;**

**}**

**int main() {**

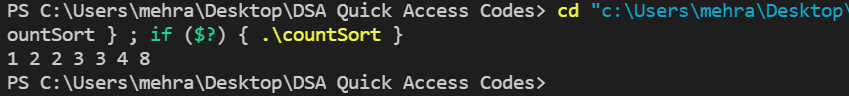
**int array[] = {4, 2, 2, 8, 3, 3, 1};**

**int n = sizeof(array) / sizeof(array[0]);**

**countSort(array, n);**

**printArray(array, n);**

**}**

****

**Radix Sort**

**#include <iostream>**

**using namespace std;**

**int getMax(int array[], int n) {**

**int max = array[0];**

**for (int i = 1; i < n; i++)**

**if (array[i] > max)**

**max = array[i];**

**return max;**

**}**

**void countingSort(int array[], int size, int place) {**

**const int max = 10;**

**int output[size];**

**int count[max];**

**for (int i = 0; i < max; ++i)**

**count[i] = 0;**

**for (int i = 0; i < size; i++)**

**count[(array[i] / place) % 10]++;**

**for (int i = 1; i < max; i++)**

**count[i] += count[i - 1];**

**for (int i = size - 1; i >= 0; i--) {**

**output[count[(array[i] / place) % 10] - 1] = array[i];**

**count[(array[i] / place) % 10]--;**

**}**

**for (int i = 0; i < size; i++)**

**array[i] = output[i];**

**}**

**void radixsort(int array[], int size) {**

**// Get maximum element**

**int max = getMax(array, size);**

**for (int place = 1; max / place > 0; place \*= 10)**

**countingSort(array, size, place);**

**}**

**void printArray(int array[], int size) {**

**int i;**

**for (i = 0; i < size; i++)**

**cout << array[i] << " ";**

**cout << endl;**

**}**

**int main() {**

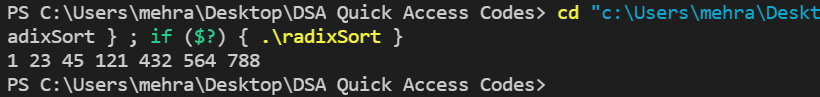
**int array[] = {121, 432, 564, 23, 1, 45, 788};**

**int n = sizeof(array) / sizeof(array[0]);**

**radixsort(array, n);**

**printArray(array, n);**

**}**

****