**Quick sort**

**Q1. Perform Quick sort on the following list:**

**{3, 1, 4, 5, 9,2,6,5}**

Algorithm quicksort(p,q)

{

if(p<q)then

{

j=partition(a,p,q+1);

quicksort(p,j-1);

quicksort(j+1,q);

}

}

#include <iostream>

using namespace std;

int partition(int arr[], int low, int high) {

int pivot = arr[low];

int i = low, j = high;

while (i < j) {

while (arr[i] <= pivot && i < high) i++;

while (arr[j] > pivot) j--;

if (i < j) swap(arr[i], arr[j]);

}

swap(arr[low], arr[j]);

return j;

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int p = partition(arr, low, high);

quickSort(arr, low, p - 1);

quickSort(arr, p + 1, high);

}

}

int main() {

int arr[] = {3, 1, 4, 5, 9, 2, 6, 5};

int n = sizeof(arr) / sizeof(arr[0]);

quickSort(arr, 0, n - 1);

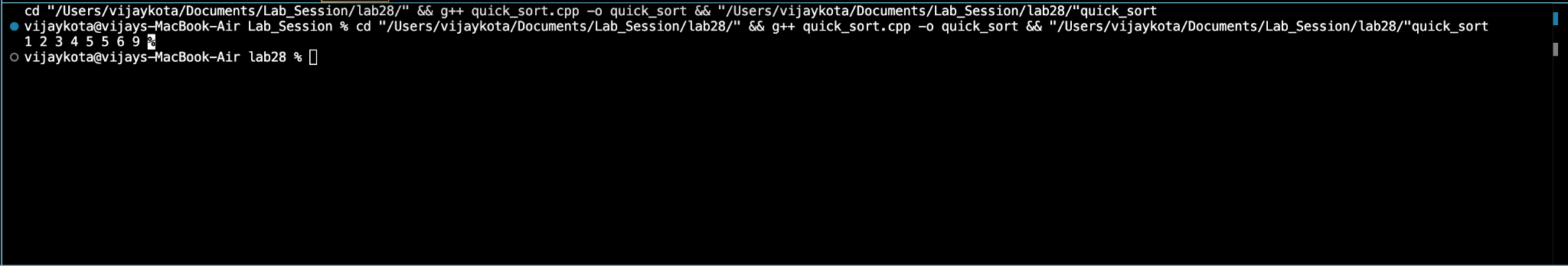
for (int i = 0; i < n; i++) {

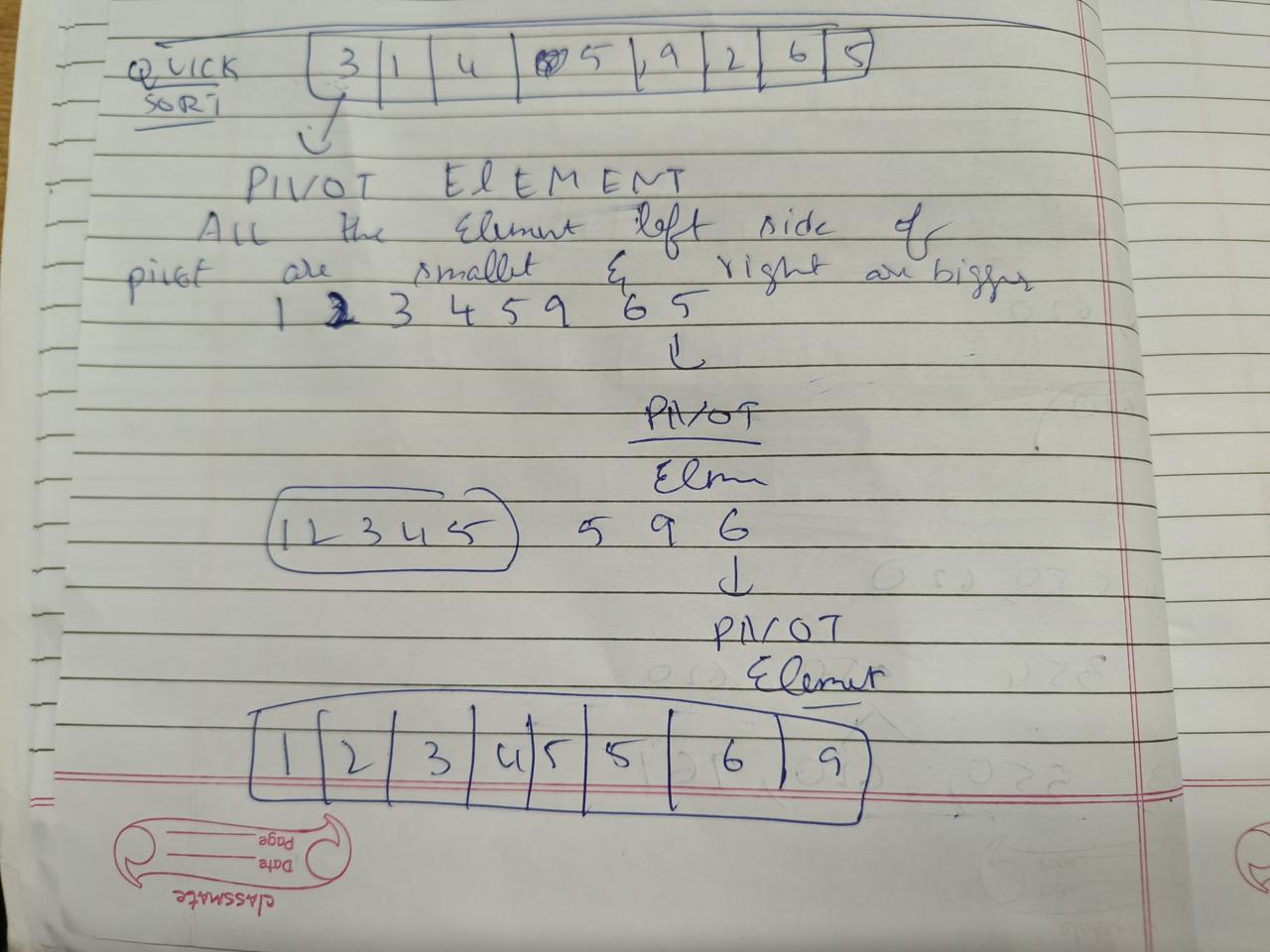
cout << arr[i] << " ";

}

return 0;

}





**Merge sort**

**Q2. Sort the following elements using merge sort:**

**{410, 385, 279, 752,451,523, 961, 354, 550, 620}**

Algorithm mergesort(low,high)

{

If (low<high)then

{

mid=[(low+high)/2];

mergesort(low,mid);

mergesort(mid+1,high);

merge(low,mid,high);

}

}

#include <iostream>

using namespace std;

void merge(int arr[], int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int L[n1], R[n2];

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

L[i] = arr[left + i];

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

R[i] = arr[mid + 1 + i];

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

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

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

int main() {

int arr[] = {410, 385, 279, 752, 451, 523, 961, 354, 550, 620};

int n = sizeof(arr) / sizeof(arr[0]);

mergeSort(arr, 0, n - 1);

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

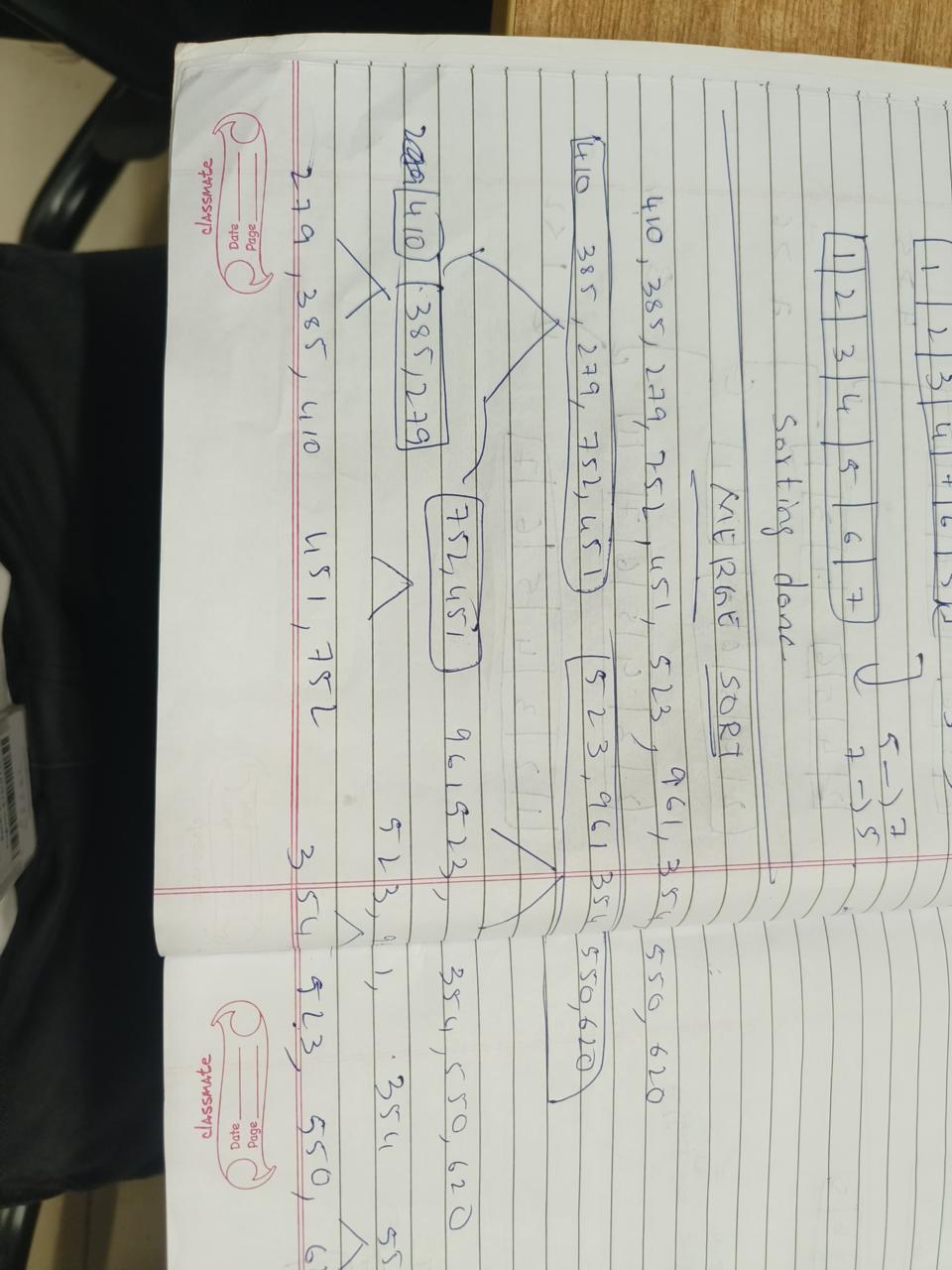
cout << arr[i] << " ";

}

return 0;

}



****

**Insertion Sort**

**Q3. Perform insertion sort:**

**{5, 6, 2, 4, 7, 3, 1}**

Algorithm insertionSort(A, n)

{

for i = 1 to n-1 do

{

key = A[i];

j = i - 1;

while j >= 0 and A[j] > key do

{

A[j + 1] = A[j];

j = j - 1;

}

A[j + 1] = key;

}

}

#include <iostream>

using namespace std;

void insertionSort(int arr[], int n) {

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j--;

}

arr[j + 1] = key;

}

}

int main() {

int arr[] = {5, 6, 2, 4, 7, 3, 1};

int n = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, n);

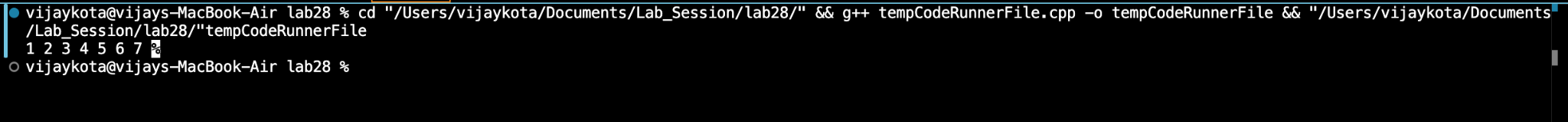
for (int i = 0; i < n; i++) {

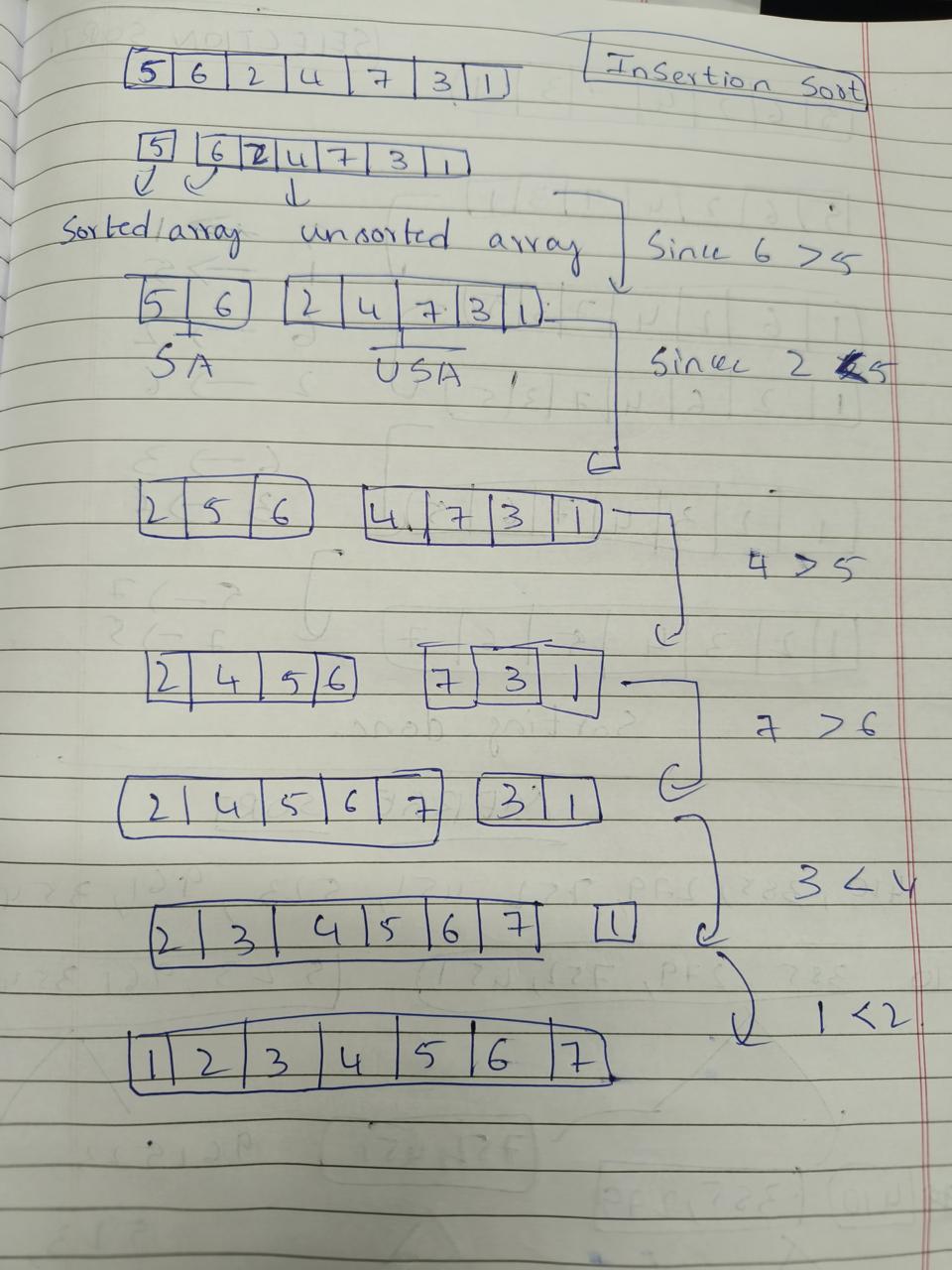
cout << arr[i] << " ";

}

return 0;

}





**Selection Sort**

**Q4. Perform selection sort:**

**{5, 6, 2, 4, 7, 3, 1}**

Algorithm selectionSort(A, n)

{

for i = 0 to n-1 do

{

minIndex = i;

for j = i+1 to n-1 do

{

if A[j] < A[minIndex] then

minIndex = j;

}

if minIndex != i then

{

swap(A[i], A[minIndex]);

}

}

}

#include <iostream>

using namespace std;

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

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

int minIndex = i;

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

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

swap(arr[i], arr[minIndex]);

}

}

}

int main() {

int arr[] = {5, 6, 2, 4, 7, 3, 1};

int n = sizeof(arr) / sizeof(arr[0]);

selectionSort(arr, n);

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

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

}

return 0;

}

