**Algorithms**

**Laboratory Task-2**

**Submission Deadline** – As announced in the class

**Submission Guidelines**-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
* Must submit within the given deadline in VUES to the section named Lab Tak-2
* Must include resources for all the section named ‘Code’ and ‘Output (screenshot)’ in the table.

|  |
| --- |
| Question-1 – **Implement Merge Sort** |
| **Pseudocode** |
| **Code**  **#include <iostream>**  **using namespace std;**  **void merge(int arr[], int l, int m, int r) {**  **int i = l;**  **int j = m + 1;**  **int k = l;**  **/\* create temp array \*/**  **int temp[5];**  **while (i <= m && j <= r) {**  **if (arr[i] <= arr[j]) {**  **temp[k] = arr[i];**  **i++;**  **k++;**  **} else {**  **temp[k] = arr[j];**  **j++;**  **k++;**  **}**  **}**  **/\* Copy the remaining elements of first half, if there are any \*/**  **while (i <= m) {**  **temp[k] = arr[i];**  **i++;**  **k++;**  **}**  **/\* Copy the remaining elements of second half, if there are any \*/**  **while (j <= r) {**  **temp[k] = arr[j];**  **j++;**  **k++;**  **}**  **/\* Copy the temp array to original array \*/**  **for (int p = l; p <= r; p++) {**  **arr[p] = temp[p];**  **}**  **}**  **/\* l is for left index and r is right index of the**  **sub-array of arr to be sorted \*/**  **void mergeSort(int arr[], int l, int r) {**  **if (l < r) {**  **// find midpoint**  **int m = (l + r) / 2;**  **// recurcive mergesort first and second halves**  **mergeSort(arr, l, m);**  **mergeSort(arr, m + 1, r);**  **// merge**  **merge(arr, l, m, r);**  **}**  **}**  **int main() {**  **int myarray[5];**  **//int arr\_size = sizeof(myarray)/sizeof(myarray[0]);**  **int arr\_size = 5;**  **cout << "Enter 5 integers in any order: " << endl;**  **for (int i = 0; i < 5; i++) {**  **cin >> myarray[i];**  **}**  **cout << "Before Sorting" << endl;**  **for (int i = 0; i < 5; i++) {**  **cout << myarray[i] << " ";**  **}**  **cout << endl;**  **mergeSort(myarray, 0, (arr\_size - 1)); // mergesort(arr,left,right) called**  **cout << "After Sorting" << endl;**  **for (int i = 0; i < 5; i++) {**  **cout << myarray[i] << " ";**  **}**  **return 0;**  **}** |
| **Output (Screenshot)** |

|  |  |
| --- | --- |
| Question-2 – **Implement Counting Sort.** | |
| **Description**  Let us understand it with the help of an example.  For simplicity, consider the data in the range 0 to 9.  Input data: 1, 4, 1, 2, 7, 5, 2  1) Take a count array to store the count of each unique object.  Index: 0 1 2 3 4 5 6 7 8 9  Count:0 2 2 0 1 1 0 1 0 0  2) Modify the count array such that each element at each index  stores the sum of previous counts.  Index: 0 1 2 3 4 5 6 7 8 9  Count: 0 2 4 4 5 6 6 7 7 7  The modified count array indicates the position of each object in the output sequence.  3) Output each object from the input sequence followed by decreasing its count by 1.  Process the input data: 1, 4, 1, 2, 7, 5, 2. Position of 1 is 2. Put data 1 at index 2 in output. Decrease count by 1 to place next data 1 at an index 1 smaller than this index. | |
| **Pseudocode**  CountingSort(A)  for i = 0 to k do  c[i] = 0  //Storing Count of each element  for j = 0 to n do  c[A[j]] = c[A[j]] + 1  // Change C[i] such that it contains actual position of these elements in output array  for i = 1 to k do  c[i] = c[i] + c[i-1]  //Build Output array from C[i]  for j = n-1 down to 0 do  B[ c[A[j]]-1 ] = A[j]  c[A[j]] = c[A[j]] - 1  end func | |
| **Code** | |
| **Pseudocode**  CountingSort(A)  for i = 0 to k do  c[i] = 0  //Storing Count of each element  for j = 0 to n do  c[A[j]] = c[A[j]] + 1  // Change C[i] such that it contains actual position of these elements in output array  for i = 1 to k do  c[i] = c[i] + c[i-1]  //Build Output array from C[i]  for j = n-1 down to 0 do  B[ c[A[j]]-1 ] = A[j]  c[A[j]] = c[A[j]] - 1  end func |
| **Code**  **#include<iostream>**  **using namespace std;**  **void show (int \*arr1, int size1)**  **{**  **for(int i = 0; i<size1; i++)**  **cout << arr1[i] << " ";**  **cout << endl;**  **}**  **void swapping(int &a, int &b)**  **{**  **int temp;**  **temp = a;**  **a = b;**  **b = temp;**  **}**  **void CountingSort(int \*arr1, int size1)**  **{**  **for(int i = 0; i<size1; i++) {**  **int swaps = 0;**  **for(int j = 0; j<size1-i-1; j++) {**  **if(arr1[j] > arr1[j+1]) {**  **swapping(arr1[j], arr1[j+1]);**  **swaps = 1;**  **}**  **}**  **if(!swaps)**  **break;**  **}**  **}**  **int main()**  **{**  **int n;**  **cout << "Enter the number of elements: ";**  **cin >> n;**  **int arr[n];**  **cout << "Enter the elements:" << endl;**  **for(int i = 0; i<n; i++)**  **{**  **cin >> arr[i];**  **}**  **cout << "Before Sorting: ";**  **show(arr, n);**  **CountingSort(arr, n);**  **cout << "After Sorting: ";**  **show(arr, n);**  **}** |
| **Output (Screenshot)** | |