Assignment 3

Algorithm:-

Divide and Conquer Strategy

Using the Divide and Conquer technique, we divide a problem into subproblems. When the solution to each subproblem is ready, we 'combine' the results from the subproblems to solve the main problem.

Suppose we had to sort an array `A`. A subproblem would be to sort a sub-section of this array starting at index `p` and ending at index `r`, denoted as `A[p..r]`.

Divide

If q is the half-way point between p and r, then we can split the subarray `A[p..r]` into two arrays `A[p..q]` and `A[q+1, r]`.

Conquer

In the conquer step, we try to sort both the subarrays `A[p..q]` and `A[q+1, r]`. If we haven't yet reached the base case, we again divide both these subarrays and try to sort them.

Combine

When the conquer step reaches the base step and we get two sorted subarrays `A[p..q]` and `A[q+1, r]` for array `A[p..r]`, we combine the results by creating a sorted array `A[p..r]` from two sorted subarrays `A[p..q]` and `A[q+1, r]`.

Merge Sort Complexity

Time Complexity

Best Case Complexity: `O(n*log n)`

Worst Case Complexity: `O(n*log n)`

Average Case Complexity: `O(n*log n)`

Space Complexity

The space complexity of merge sort is `O(n)`.

Code:-

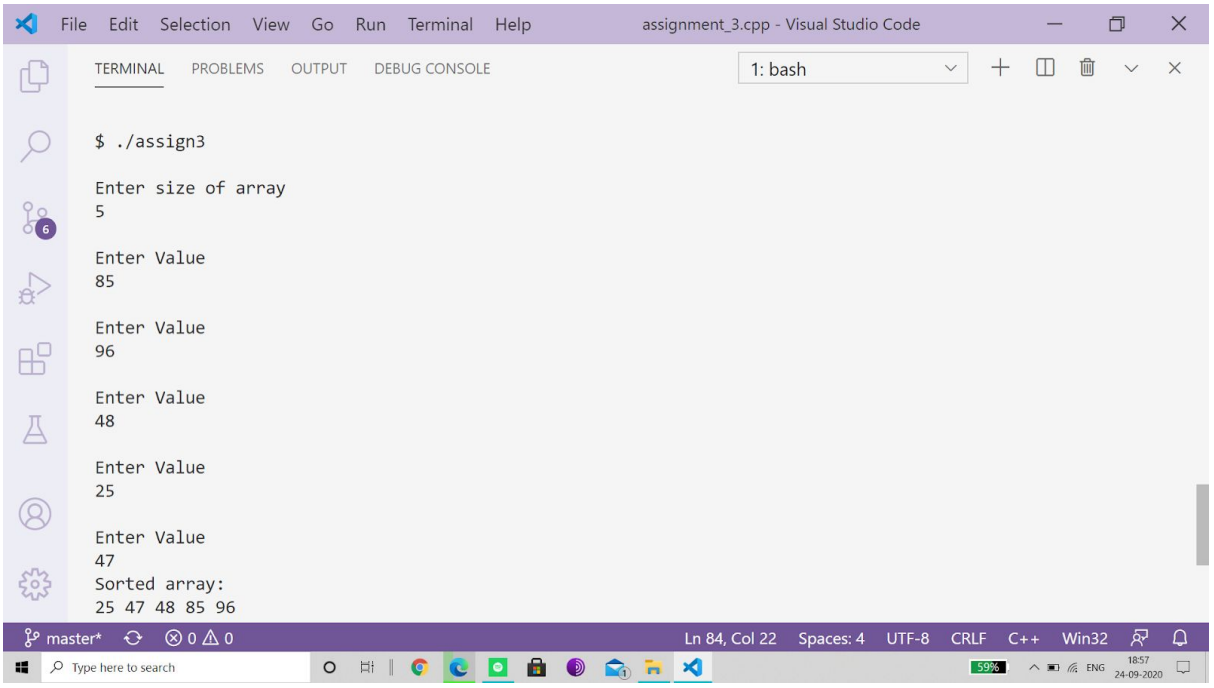
```
 // Merge sort in C++
```

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

// Merge two subarrays L and M into arr
void merge(int arr[], int p, int q, int r) {

  // Create L ← A[p..q] and M ← A[q+1..r]
  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];

  // Maintain current index of sub-arrays and main array
```

```
  int i, j, k;
  i = 0;
  j = 0;
  k = p;

  // Until we reach either end of either L or M, pick larger among
  // elements L and M and place them in the correct position at A[p..r]
  while (i < n1 && j < n2) {
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }

  // When we run out of elements in either L or M,
  // pick up the remaining elements and put in A[p..r]
  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}

// Divide the array into two subarrays, sort them and merge them
void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    // m is the point where the array is divided into two subarrays
    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    // Merge the sorted subarrays
```

```
    merge(arr, l, m, r);
  }
}

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

// Driver program
int main() {

  cout<<"\nEnter size of array\n";
  int size;
  cin>>size;
  int arr[size];
  for(int i=0;i<size;i++)
{
    cout<<"\nEnter Value\n";
    cin>>arr[i];
}
  mergeSort(arr, 0, size - 1);

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

Output:-


File Edit Selection View Go Run Terminal Help
assignment_3.cpp - Visual Studio Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: bash
$ ./assign3
Enter size of array
5
Enter Value
85
Enter Value
96
Enter Value
48
Enter Value
25
Enter Value
47
Sorted array:
25 47 48 85 96
master*
Ln 84, Col 22 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search