**Sri Sri University, Cuttack, Odisha.**

**Faculty of Science**

|  |  |
| --- | --- |
| **Program: B.Sc. – Computer Science, Data Science, & Environmental Science**  **(2020-23 Batch)**  **Subject Code/Subject Name: Data Structure Laboratory**  **Assignment –XII** | |
| **Full Name of the Student:** | VINAYAK SANJAY CHAVAN |
| **Full Roll Number:** | BCS-011 |
| **Program:** | B.Sc. (Computer Sc.) / B.Sc. (Data Sc.) / B.Sc. (Env. Sc.) |
| **Date:** | 10th May, 2021 (10.00 AM – 12.00 Noon) |
| **Signature** |  |

|  |  |
| --- | --- |
| **All Questions are compulsory** | **Total Marks: 60** |

|  |  |
| --- | --- |
| **Question (s)** | **Maximum Marks** |
| 1. Write a function to implement recursive quick sort algorithm and using this function sort an array of integers.   **Your code:** **#include <iostream>**  **using namespace std;**  **// function to swap elements**  **void swap(int \*a, int \*b) {**  **int t = \*a;**  **\*a = \*b;**  **\*b = t;**  **}**  **// function to print the array**  **void printArray(int array[], int size){**  **int i;**  **for (i = 0; i < size; i++){**  **cout << array[i] << " ";**  **}**  **cout << endl;**  **}**  **// function to rearrange array (find the partition point)**  **int partition(int array[], int low, int high) {**    **// select the rightmost element as pivot**  **int pivot = array[high];**    **// pointer for greater element**  **int i = (low - 1);**  **// traverse each element of the array**  **// compare them with the pivot**  **for (int j = low; j < high; j++) {**  **if (array[j] <= pivot) {**    **// if element smaller than pivot is found**  **// swap it with the greater element pointed by i**  **i++;**    **// swap element at i with element at j**  **swap(&array[i], &array[j]);**  **}**  **}**    **// swap pivot with the greater element at i**  **swap(&array[i + 1], &array[high]);**    **// return the partition point**  **return (i + 1);**  **}**  **void quickSort(int array[], int low, int high) {**  **if (low < high) {**    **// find the pivot element such that**  **// elements smaller than pivot are on left of pivot**  **// elements greater than pivot are on righ of pivot**  **int pi = partition(array, low, high);**  **// recursive call on the left of pivot**  **quickSort(array, low, pi - 1);**  **// recursive call on the right of pivot**  **quickSort(array, pi + 1, high);**  **}**  **}**  **int main() {**  **int element[] = {33, 1, 02, 43, 54, 11, 44};**  **int n = sizeof(element) / sizeof(element[0]);**    **cout << "Unsorted Array: \n";**  **printArray(element, n);**    **// perform quicksort on data**  **quickSort(element, 0, n - 1);//low =0**    **cout << "Sorted array in ascending order: \n";**  **printArray(element, n);**  **}**  **Screenshot of output:** | 20 |
| 1. Implement recursive mergesort using an array of fixed size and hence sort an array of double numbers using this function.   **Your code: #include<iostream>**  **using namespace std;**  **void swapping(int &a, int &b) {**  **int temp;**  **temp = a;**  **a = b;**  **b = temp;**  **}**  **void display(int \*array, int size) {**  **for(int i = 0; i<size; i++)**  **cout << array[i] << " ";**  **cout << endl;**  **}**  **void merge(int \*array, int l, int m, int r) {**  **int i, j, k, nl, nr;**  **nl = m-l+1;//L+1**  **nr = r-m;//low=mid+1 right=ub**    **int larr[nl], rarr[nr];**    **for(i = 0; i<nl; i++){**  **larr[i] = array[l+i];**  **}**  **for(j = 0; j<nr; j++){**  **rarr[j] = array[m+1+j];**  **}**  **i = 0; j = 0; k = l;**  **//marge temp and arr**  **while(i < nl && j<nr) {**  **if(larr[i] <= rarr[j]) {**  **array[k] = larr[i];**  **i++;**  **}**  **else{**  **array[k] = rarr[j];**  **j++;**  **}**  **k++;**  **}**  **while(i<nl) {**  **array[k] = larr[i];**  **i++;**  **k++;**  **}**  **while(j<nr) {**  **array[k] = rarr[j];**  **j++;**  **k++;**  **}**  **}**  **void mergeSort(int \*array, int l, int r) {**  **int m;**  **if(l < r) {**  **int m = l+(r-l)/2;**  **// Sort 1st and 2nd arrays**  **mergeSort(array, l, m);**  **mergeSort(array, m+1, r);**  **merge(array, l, m, r);**  **}**  **}**  **int main() {**  **int n;**  **cout << "Enter the number of elements: ";**  **cin >> n;**  **int arr[n];**  **cout << "Enter elements:" << endl;**  **for(int i = 0; i<n; i++) {**  **cin >> arr[i];**  **}**  **cout << "Array before Sorting: ";**  **display(arr, n);**  **mergeSort(arr, 0, n-1); //(n-1) for last index**  **cout << "Array after Sorting: ";**  **display(arr, n);**  **}**  **Screenshot of output:** | 20 |
| 1. Implement the radix sort to sort decimal numbers.   **Your code:** **//implementation of Radix Sort**  **#include <iostream>**  **using namespace std;**  **// A utility function to get maximum value in arr[]**  **int max\_val(int arr[], int n){**  **int max = arr[0];**  **for (int i = 1; i < n; i++)**  **if (arr[i] > max)**  **max = arr[i];**  **return max;**  **}**  **// A function to do counting sort of arr[] according to**  **// the digit represented by exp.**  **void countSort(int arr[], int n, int exp){**  **int output[n]; // output array**  **int i, count[10] = { 0 };**  **// Store count of occurrences in count[]**  **for (i = 0; i < n; i++)**  **count[(arr[i] / exp) % 10]++;**  **// Change count[i] so that count[i] now contains actual**  **// position of this digit in output[]**  **for (i = 1; i < 10; i++)**  **count[i] += count[i - 1];**  **// Build the output array**  **for (i = n - 1; i >= 0; i--) {**  **output[count[(arr[i] / exp) % 10] - 1] = arr[i];**  **count[(arr[i] / exp) % 10]--;**  **}**  **// Copy the output array to arr[], so that arr[] now**  **// contains sorted numbers according to current digit**  **for (i = 0; i < n; i++)**  **arr[i] = output[i];**  **}**  **// The main function to that sorts arr[] of size n using**  **// Radix Sort**  **void radixsort(int arr[], int n){**  **// Find the maximum number to know number of digits**  **int m = max\_val(arr, n);**  **for (int exp = 1; m / exp > 0; exp \*= 10)**  **countSort(arr, n, exp);**  **}**  **// A utility function to display an array**  **void display(int arr[], int n){**  **for (int i = 0; i < n; i++)**  **cout << arr[i] << " ";**  **}**  **// Driver Code**  **int main(){**  **int arr[] = {23, 12, 90, 45, 43, 54, 11, 0};**  **int n = sizeof(arr) / sizeof(arr[0]);**    **// Function Call**  **radixsort(arr, n);**  **display(arr, n);**  **return 0;**  **}**  **Screenshot of output:** | 20 |