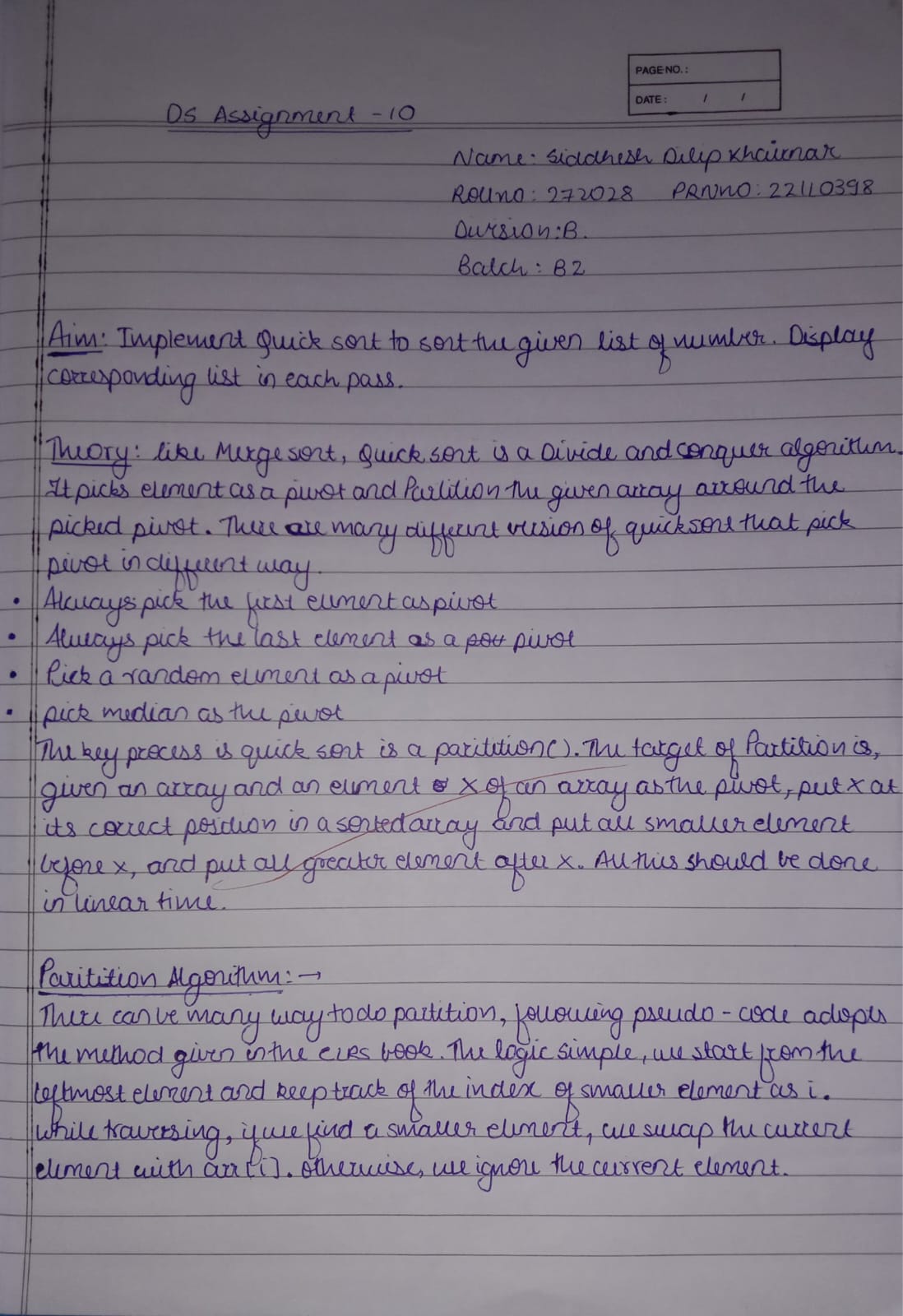
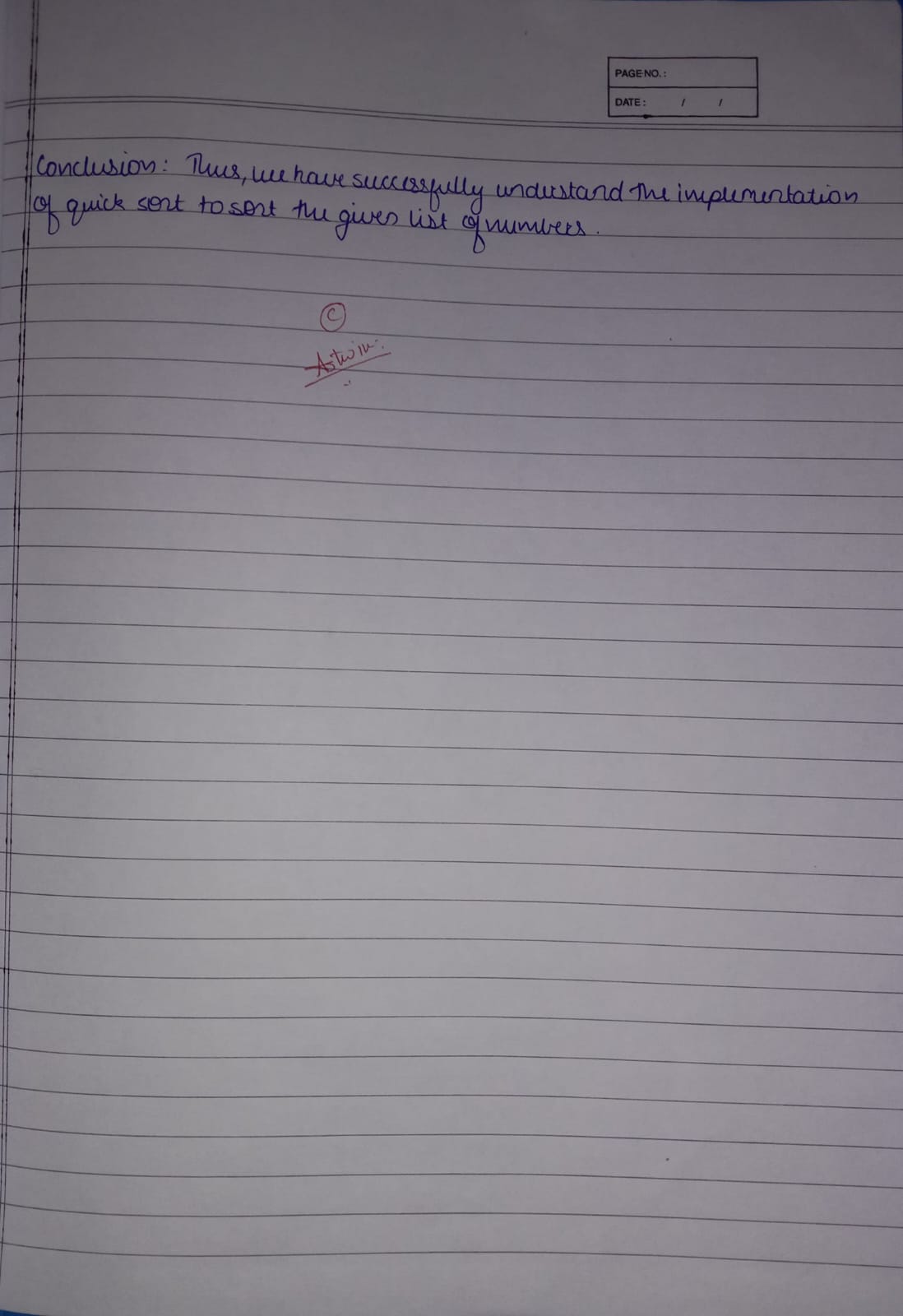
|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: SY | Division: B | | Roll No: 272028 |
| Semester: III | | Academic Year: 2022-2023 | |
| Subject Name & Code: Data Structure, ADUA21202 | | | |
| Title of Assignment:Sort the data in ascending order using Quick sort (Display corresponding list in each pass). | | | |

**Assignment No.- 10**





**Program:**

// Quick sort in C++

#include <iostream>

using namespace std;

int data[] = {8, 7, 6, 1, 0, 9, 2};

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

// 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]);

    }

    printArray(data,n);

  }

  // 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);

  }

}

// Driver code

int main() {

  //int data[] = {8, 7, 6, 1, 0, 9, 2};

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

  cout << "Unsorted Array: \n";

  printArray(data, n);

  cout<<"Steps"<<endl;

  // perform quicksort on data

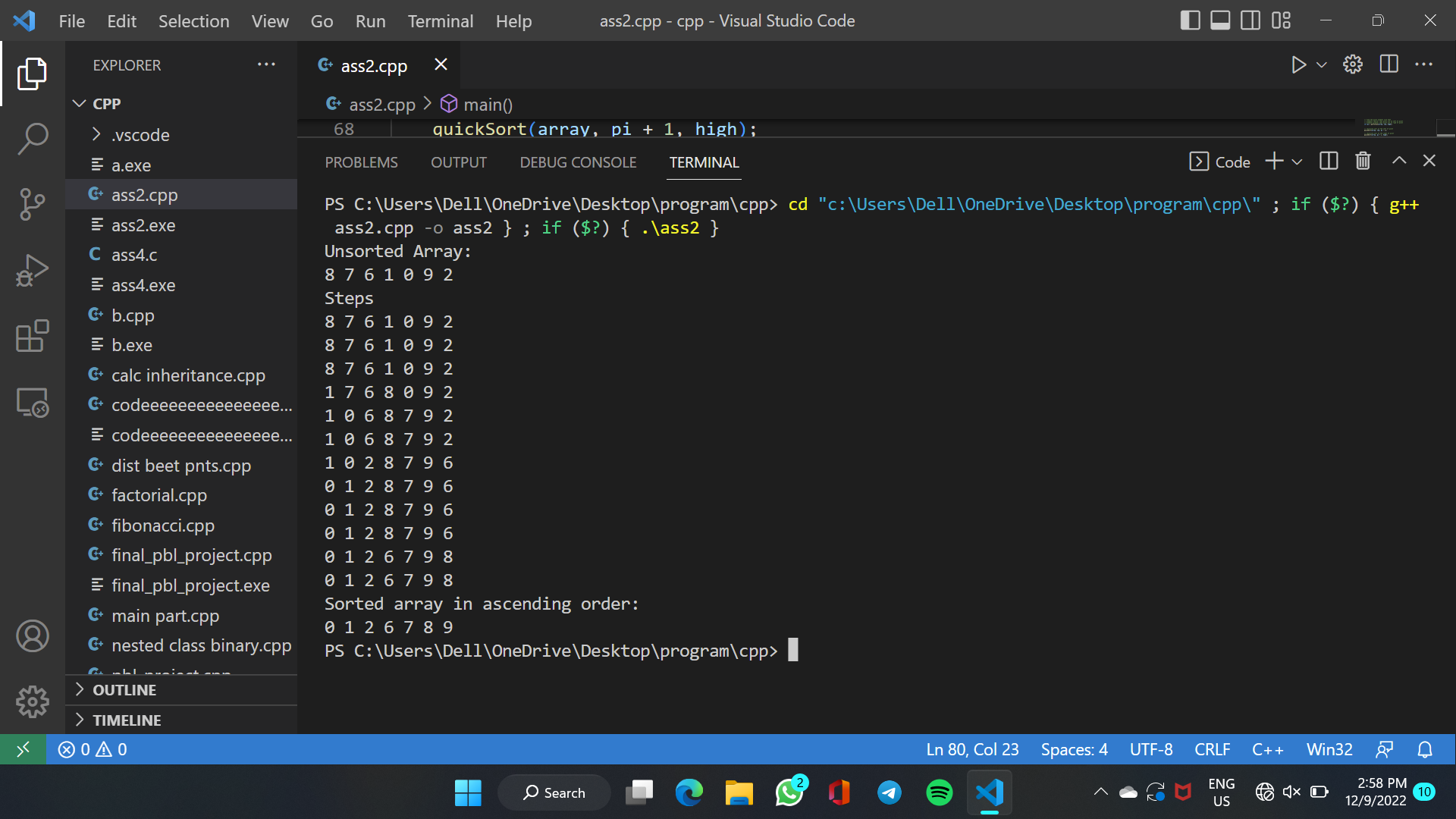
  quickSort(data, 0, n - 1);

  cout << "Sorted array in ascending order: \n";

  printArray(data, n);

}

**Output:**

****