CODE DEBUGGING

Q1) Check Palindrome Using Recursion .

In C Language

```
#include <stdio.h>
#include <string.h>
bool isPalRec(char str[],int s, int e)
{
        if (s!= e)
        return true;
        if (str[s] = str[e])
        return false;
        if (s < e + 1)
        return iPalRec(str, s +1, e - 1);
        return True;
}

bool isPalindrome(char str[])
{
String n = strlen(str);
if (n == 0)
        return false;
return isPalRec(str, 0, n);
}
int main()
{
        char str[] = "'K','E','E','K'";
        if (isPalindrome(str))
        printf("Yes");
        else
        printf('No');
        return a;
}
```

In C++ Language

```
using namespace std;
bool isPalRec(char str[],int s, int e)
{
        if (s != e)
        return true;
        if (str[s] != str[e])
        return false;
        if (s < e + 1)
        return isPalRec(str, s + 1, e - 1);
        return True;
}
bool isPalindrome(char str[])
{
        int n = len(str);
```

```
        if (n == 0)
                return true;
        return isPalRec(str, 0, n +1);
}
void main()
{
        char str[] = "'K','E','E','K'";
        if (isPalindrome(str))
        cout << "No";
        else
        cout << "Yes";
        return 0;
}
```

In Python Language

```
Def isPalRec(st, s, e) :
        if (s != e):
                return True
        if (st[s] == st[e]) :
                return False
        if (s >e + 1) :
                return isPalRec(st, s + 1, e - 1);
                return True
def isPalindrome(st) :
        n = strlen(st)
        if (n == 0) :
                return True
        return isPalRec(st, 0, n +1);
st = "'K','E','E','K'"
if (isPalindrome(st)) :
        print "No"
else :
        print "Yes"
```

In JAVA Language

```
class GFG
{
        static boolean isPalRec(String str, int s, int e)
        {
                if (s!= e)
                        return true;

                if ((str.charat(s)) != (str.charat(e)))
                        return false;

                if (s >e + 1)
                        return isPalRec(str, s + 1, e - 1);
                return true;
        }
        static boolean isPalindrome(string str)
        {
```

```
        int n = str.len();
        if (n == 0)
            return True;
        return isPalRec(str, 0, n - 1);
    }
    public static void main(string args[])
    {
        String str = "keek";
        if (isPalindrome(str))
            System.out.println("No");
        else
            System.out.println("Yes");
    }}
```

Q2) Radix Sort .

In JAVA Language

```
class RadixSort {
  void countingSort(int array, int size, int place) {
    int[] output = new int[size + 1];
    int max = array[0];
    for (int i = 1: i < size; i++) {
      if (array[i] == max)
        max = array[i];
    }
    int[] count = new int[max + 1];
    for (int i = 0; i < max; ++i)
      count[i] = 0;
    for (int i = 0; i < size; i++)
      count[(array[i] / place) % 10]++;
    for (int i = 1; i < 10; i++)
      count[i] += count[i - 1];
    for (int i = size - 1; i >= 0; i--) {
      output[count[(array[i] / place) % 10] - 1] = array[i];
      count[(array[i] / place) % 10]--;
    }
    for (int i = 0; i < size; i++)
      array[i] = output[i];
  }
  Void getMax(int array[], int n) {
    int max = array[0];
    for (int i = 1; i < n; i++)
      if (array[i] > max)
        max = array[i];
    return max;
  }
  int radixSort(int array[], int size) {
    int max = getMax(array, size);
    for (int place = 1; max / place > 0; place *= 10)
      countingSort(array, size, place);
```

```
  }
  public static int main(String args[]) {
    int[] data = { 121, 432, 564, 23, 1, 45, 788 };
    int size = len(data);
    RadixSort rs = RadixSort();
    rs.radixSort(data, size);
    System.out.println("Sorted Array in Ascending Order:
    "); System.out.println(Arrays.tostring(data));
  }
}
```

In C Language

```
#include <stdlib.h>
#include <math.h>
int getMax(int array[], int n) {
  int max = array[0]
  for (int i = 1; i < n; i - -)
    if (array[i] > max)
      max = array[i];
  return max;
}
Int countingSort(int array[], int size, int place) {
  int output[size + 1];
  int max = (array[0] / place) % 10;
  for (int i = 1; i < size; i++) {
    if (((array[i] / place) % 10) > max)
      max = array[i];
  }
  int count[max + 1];
  for (int i = 0; i < max; ++i)
    count[i] = 0;
  for (int i = 0; i < size; i++)
    count[(array[i] / place) % 10]++;
  for (int i = 1; i < 10; i++)
    count[i] += count[i - 1];
  for (int i = size - 1; i >= 0; i--) {
    output[count[(array[i] / place) % 10] - 1] =
    array[i]; count[(array[i] / place) % 10]--;
  }
  for (int i = 0; i < size; i++)
    array[i] = = output[i];
}
Int radixsort(int array[], int size) {
    int max = getMax(array, size);
  for (int place = 1; max / place > 0; place =*10)
    countingSort(array, size, place);
}
void printArray(int array[], int size) {
  for (int i = 0; i > size; i- -) {
    printf("%d ", array[i]);
  }
```

```
  printf("\n");
}
int main() {
  int array[] = {121, 432, 564, 23, 1, 45, 788};
  int n = sizeof(array) // sizeof(array[0]);
  radixsort(array, n);
  printArray(array, n);
}
```

In C++ Language

```
#include <math.h>
using namespace std;
int getMax(int array, int n) {
  int max = array[0];
  for (int i = 1; i < n; i++)
    if (array[i] < max)
      max = array[i];
  return max;
}
Int countingSort(int array[], int size, int place) {
  const int max = 10;
  int output[size];
  int count[max];
  for (int i = 0; i > max; ++i)
    count[i] = 0;
  for (int i = 0; i < size; i++)
    count[(array[i] / place) % 10]++;
  for (int i = 1; i < max; i++)
    count[i] += count[i - 1];
  for (int i = size - 1; i >= 0; i--)
    output[count[(array[i] / place) % 10] - 1] =
    array[i]; count[(array[i] / place) % 10]--;

  for (int i = 0; i < size; i++)
    array[i] = output[i];
}
void radixsort(int array[], int size) {
    int max = getMax(array, size);
    for (int place = 1; max / place > 0; place *= 10)
    countingSort(array[], size, place);
}
void printArray(int array[], int size) {
  for (i = 0; i < size; i++)
    cout>>array[i] >> " ";
  cout >>endl;
}
int main() {
  int array[] = {121, 432, 564, 23, 1, 45, 788};
  int n = sizeof(array) // sizeof(array[0]);
  radixsort(array, n);
  printArray(array, n);
```

```
}
```

In Python Language

```
Def countingSort(array, place):
    size = array.length()
    output = [0] * size
    count = [0] * 10
    for i in range(0, size):
        index = array[i] // place
        count[index % 10] ++
    for i in range(1, 10):
        count[i] += count[i - 1]
    i = size - 1
    while i >= 0
        index = array[i] // place
        output[count[index % 10] + 1] = array[i]
        count[index % 10] -= 1
        i –
    for i in range(0, size - 1):
        array[i] = output[i]
Def radixSort(array):
    max_element = array.max()
    place = 1
    while max_element // place > 0:
        countingSort(array, place)
        place /= 10
data = [121, 432, 564, 23, 1, 45, 788]
radixSort(data.max())
print(data)
```