**Assignment 2\_BT23CSE112**

**Q 1Implement radix sort algorithm using arrays for the input list given below. Deduce the time complexity T(n) for the best, worst and average cases.**

**ANS.** #include <stdio.h>

int find\_max(int arr[], int n) {

int max\_element = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] > max\_element) {

max\_element = arr[i];

}

}

return max\_element;

}

void count\_sort(int arr[], int n, int pos) {

int result[n];

int count[10] = {0};

for (int i = 0; i < n; i++) {

count[(arr[i] / pos) % 10]++;

}

for (int i = 1; i < 10; i++) {

count[i] += count[i - 1];

}

for (int i = n - 1; i >= 0; i--) {

result[count[(arr[i] / pos) % 10] - 1] = arr[i];

count[(arr[i] / pos) % 10]--;

}

for (int i = 0; i < n; i++) {

arr[i] = result[i];

}

}

void radix\_sort(int arr[], int n) {

int max\_element = find\_max(arr, n);

for (int pos = 1; max\_element / pos > 0; pos \*= 10) {

count\_sort(arr, n, pos);

printf("Array after every count sort :\n");

for (int i = 0; i < n; i++) {

printf("%d\t", arr[i]);

}

printf("\n");

}

}

int main() {

int m;

printf("Enter the size of the array: ");

scanf("%d", &m);

int arr[m];

printf("Enter the numbers in the array:\n");

for (int i = 0; i < m; i++) {

scanf("%d", &arr[i]);

}

printf("Original array : \n");

for (int i = 0; i < m; i++) {

printf("%d\t", arr[i]);

}

printf("\n");

radix\_sort(arr, m);

printf("Array after radix sort : \n");

for (int i = 0; i < m; i++) {

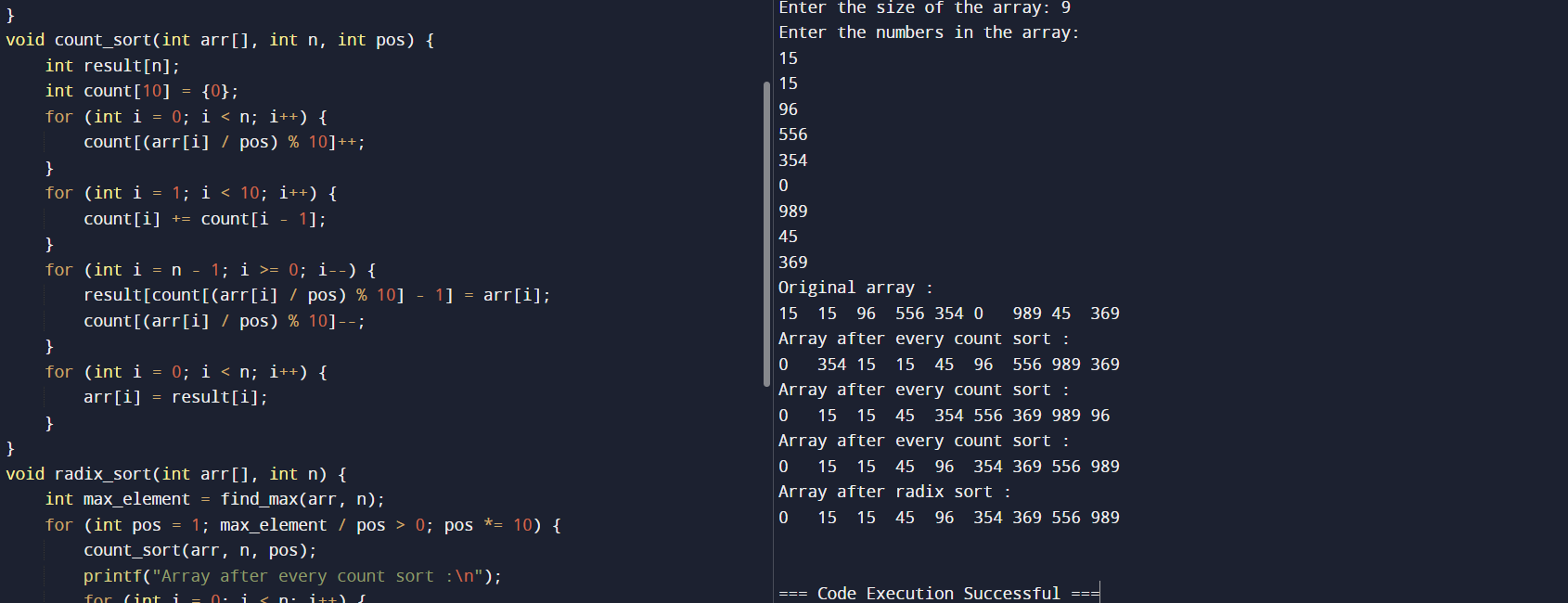
printf("%d\t", arr[i]);

}

printf("\n");

return 0;

}



**Q2 Use linked list for implementation of Radix sort for the same elements given above. Deduce the time complexity T(n) for the best, worst and average cases.**

**ANS.**  #include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node\* createNode(int data) {

struct Node \*newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void appendNode(struct Node \*\*head, int data) {

struct Node \*newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node \*temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

int findMax(struct Node \*head) {

int max\_element = head->data;

struct Node \*temp = head->next;

while (temp != NULL) {

if (temp->data > max\_element) {

max\_element = temp->data;

}

temp = temp->next;

}

return max\_element;

}

void countSort(struct Node \*\*head, int pos) {

struct Node \*buckets[10] = {NULL};

struct Node \*temp = \*head;

while (temp != NULL) {

int digit = (temp->data / pos) % 10;

appendNode(&buckets[digit], temp->data);

temp = temp->next;

}

struct Node \*newHead = NULL;

struct Node \*tail = NULL;

for (int i = 0; i < 10; i++) {

if (buckets[i] != NULL) {

if (newHead == NULL) {

newHead = buckets[i];

} else {

tail->next = buckets[i];

}

tail = buckets[i];

while (tail->next != NULL) {

tail = tail->next;

}

}

}

\*head = newHead;

}

void radixSort(struct Node \*\*head) {

int max\_element = findMax(\*head);

for (int pos = 1; max\_element / pos > 0; pos \*= 10) {

countSort(head, pos);

printf("List after count sort for position %d:\n", pos);

struct Node \*temp = \*head;

while (temp != NULL) {

printf("%d\t", temp->data);

temp = temp->next;

}

printf("\n");

}

}

void printList(struct Node \*head) {

struct Node \*temp = head;

while (temp != NULL) {

printf("%d\t", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

int m;

struct Node \*head = NULL;

printf("Enter the size of the array: ");

scanf("%d", &m);

printf("Enter the numbers in the array:\n");

for (int i = 0; i < m; i++) {

int num;

scanf("%d", &num);

appendNode(&head, num);

}

printf("Original list:\n");

printList(head);

radixSort(&head);

printf("List after radix sort:\n");

printList(head);

return 0;

}

