**DSA Lab Questions**

**LAB 2**

Q1

**Code**

#include <stdio.h>

#include <stdlib.h>

int getMax(int arr[], int n) {

    int max = arr[0];

    for (int i = 1; i < n; i++) {

        if (arr[i] > max) {

            max = arr[i];

        }

    }

    return max;

}

void countingSort(int arr[], int n, int exp) {

    int output[n];

    int count[10] = {0};

    for (int i = 0; i < n; i++) {

        count[(arr[i] / exp) % 10]++;

    }

    for (int i = 1; i < 10; i++) {

        count[i] += count[i - 1];

    }

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

        output[count[(arr[i] / exp) % 10] - 1] = arr[i];

        count[(arr[i] / exp) % 10]--;

    }

    for (int i = 0; i < n; i++) {

        arr[i] = output[i];

    }

}

void radixSort(int arr[], int n) {

    int max = getMax(arr, n);

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

        countingSort(arr, n, exp);

    }

}

void printArray(int arr[], int n) {

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main() {

    int arr[] = {136, 487, 358, 469, 570, 247, 598, 639, 205, 609};

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

    printf("Original array: ");

    printArray(arr, n);

    radixSort(arr, n);

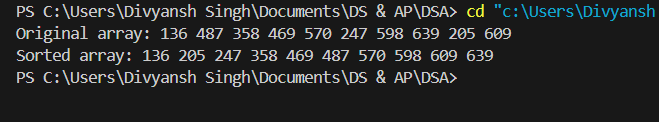
    printf("Sorted array: ");

    printArray(arr, n);

    return 0;

}

**OUTPUT**

****

Q2

**Code**

#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;

}

int getMax(struct Node\* head) {

    int max = head->data;

    struct Node\* temp = head;

    while (temp != NULL) {

        if (temp->data > max) {

            max = temp->data;

        }

        temp = temp->next;

    }

    return max;

}

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

    struct Node\* newNode = createNode(data);

    if (\*head == NULL) {

        \*head = newNode;

        return;

    }

    struct Node\* temp = \*head;

    while (temp->next != NULL) {

        temp = temp->next;

    }

    temp->next = newNode;

}

void countingSort(struct Node\*\* head, int exp) {

    struct Node\* output = NULL;

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

    struct Node\* temp = \*head;

    while (temp != NULL) {

        int bucketIndex = (temp->data / exp) % 10;

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

        temp = temp->next;

    }

    for (int i = 0; i < 10; i++) {

        if (buckets[i] != NULL) {

            if (output == NULL) {

                output = buckets[i];

            } else {

                struct Node\* temp = output;

                while (temp->next != NULL) {

                    temp = temp->next;

                }

                temp->next = buckets[i];

            }

        }

    }

    \*head = output;

}

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

    int max = getMax(\*head);

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

        countingSort(head, exp);

    }

}

void printList(struct Node\* head) {

    struct Node\* temp = head;

    while (temp != NULL) {

        printf("%d ", temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main() {

    int arr[] = {136, 487, 358, 469, 570, 247, 598, 639, 205, 609};

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

    struct Node\* head = NULL;

    for (int i = 0; i < n; i++) {

        appendNode(&head, arr[i]);

    }

    printf("Original list: ");

    printList(head);

    radixSort(&head);

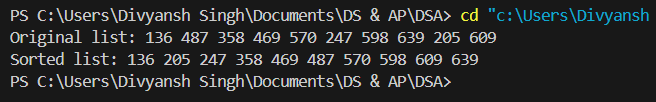
    printf("Sorted list: ");

    printList(head);

    return 0;

}

**OUTPUT**

****