Q1 : Implement radix sort algorithm using arrays :

#include<stdio.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 n;

    printf("enter the number of elements in array");

    scanf("%d",&n);

    int arr[n];

    for(int i =0; i< n ; i++){

        scanf("%d", &arr[i]);

    }

    printf("Original array: \n");

    printarray(arr, n);

    radixSort(arr, n);

    printf("Sorted array: \n");

    printarray(arr, n);

    return 0;

}

OUTPUT :

enter the number of elements in array5

1

54

23

87

45

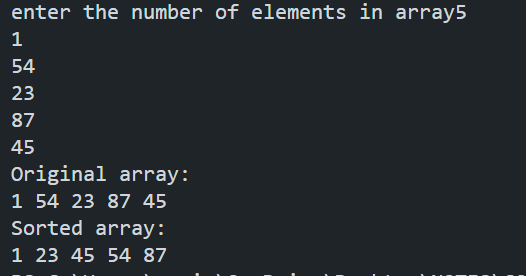
Original array:

1 54 23 87 45

Sorted array:

1 23 45 54 87

SCREENSHOT :



Q2  Implement radix sort algorithm using Linkedlist :

#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 insertNode(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 = head->data;

    struct Node\* temp = head;

    while (temp != NULL) {

        if (temp->data > max) {

            max = temp->data;

        }

        temp = temp->next;

    }

    return max;

}

int getDigit(int number, int place) {

    return (number / place) % 10;

}

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

    int max = findMax(\*head);

    int place = 1;

    while (max / place > 0) {

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

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

        struct Node\* temp = \*head;

        while (temp != NULL) {

            int digit = getDigit(temp->data, place);

            if (buckets[digit] == NULL) {

                buckets[digit] = last[digit] = temp;

            } else {

                last[digit]->next = temp;

                last[digit] = temp;

            }

            temp = temp->next;

        }

        \*head = NULL;

        struct Node\* current = NULL;

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

            if (buckets[i] != NULL) {

                if (\*head == NULL) {

                    \*head = buckets[i];

                    current = last[i];

                } else {

                    current->next = buckets[i];

                    current = last[i];

                }

            }

        }

        if (current != NULL) {

            current->next = NULL;

        }

        place \*= 10;

    }

}

void printList(struct Node\* head) {

    struct Node\* temp = head;

    while (temp != NULL) {

        printf("%d ", temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main() {

    struct Node\* head = NULL;

    insertNode(&head, 170);

    insertNode(&head, 45);

    insertNode(&head, 75);

    insertNode(&head, 90);

    insertNode(&head, 802);

    insertNode(&head, 24);

    insertNode(&head, 2);

    insertNode(&head, 66);

    printf("Original List:\n");

    printList(head);

    radixSort(&head);

    printf("Sorted List:\n");

    printList(head);

    return 0;

}

OUTPUT :

Original List:

170 45 75 90 802 24 2 66

Sorted List:

2 24 45 66 75 90 170 802

SCREENSHOT :

