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



A) #include<stdio.h>

#include<stdlib.h>

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

int max=arr[0];

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

if(arr[i]>max){

max=arr[i];

}

}

return max;

}

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

int \*count=(int \*)calloc(10,sizeof(int));

int \*output=(int \*)malloc(n\*sizeof(int));

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];

}

free(count);

free(output);

}

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

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

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

}

printf("\n");

}

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

int max=Max(arr,n);

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

countsort(arr,n,exp);

}

}

int main(){

int n;

printf("Enter the number of elements: ");

scanf("%d",&n);

int \*arr=(int \*)malloc(n\*sizeof(int));

printf("Enter %d elements: ",n);

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

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

}

printf("before sorting: \n");

printarray(arr,n);

radixsort(arr,n);

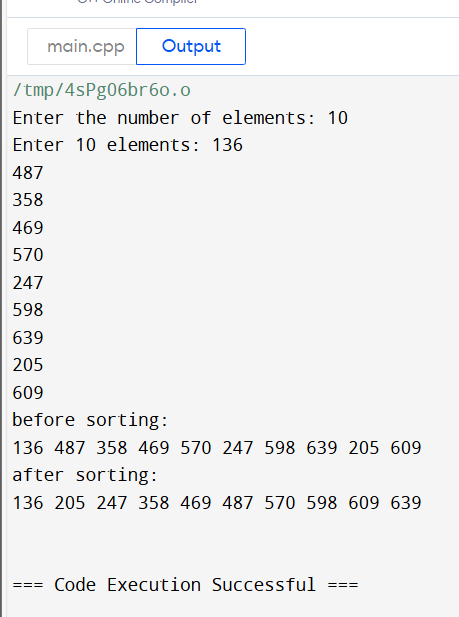
printf("after sorting: \n");

printarray(arr,n);

free(arr);

return 0;

}



(b) 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



A) #include<stdio.h>

#include<stdlib.h>

struct node {

int data;

struct node \*next;

};

typedef struct node node;

struct node \*create(node \*head, int n, int t) {

struct node \*temp, \*new = (node \*)malloc(sizeof(struct node));

int temp1 = (n / t) % 10;

new->data = n;

new->next = NULL;

if (head == NULL)

head = new;

else {

if (((head->data / t) % 10) > temp1) {

new->next = head;

head = new;

} else {

temp = head;

while (temp->next != NULL && (temp->next->data / t) % 10 <= temp1) {

temp = temp->next;

}

new->next = temp->next;

temp->next = new;

}

}

return head;

}

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

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

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

}

printf("\n");

}

void radixsort(int \*a, int n) {

int i, j, k, temp1, copy[n];

struct node \*head;

temp1 = a[0];

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

if (a[i] > temp1)

temp1 = a[i];

}

int d = 0, rem;

while (temp1 != 0) {

d++;

rem = temp1 % 10;

temp1 = temp1 / 10;

}

int t = 1;

for (i = 1; i <= d; i++) {

head = NULL;

for (j = 0; j < n; j++) {

head = create(head, a[j], t);

}

struct node \*temp = head;

for (j = 0; j < n && temp != NULL; j++) {

a[j] = temp->data;

temp = temp->next;

}

t = t \* 10;

}

}

int main() {

int n;

printf("Enter elements in linked list: ");

scanf("%d", &n);

int a[n], i;

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

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

printf("before sorting: \n");

printarray(a, n);

radixsort(a, n);

printf("after sorting: \n");

printarray(a, n);

return 0;

}

