Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

You are working as a programmer at a sports academy, and the academy holds various sports competitions regularly.

As part of the academy's system, you need to sort the scores of the participants in descending order using the Quick Sort algorithm.

Write a program that takes the scores of n participants as input and uses the Quick Sort algorithm to sort the scores in descending order. Your program should display the sorted scores after the sorting process.

Input Format

The first line of input consists of an integer n, which represents the number of scores.

The second line of input consists of n integers, which represent scores separated by spaces.

Output Format

Each line of output represents an iteration of the Quick Sort algorithm, displaying the elements of the array at that iteration.

After the iterations are complete, the last line of output prints the sorted scores in descending order separated by space.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
78 54 96 32 53
Output: Iteration 1: 78 54 96 53 32
Iteration 2: 96 54 78
Iteration 3: 78 54
Sorted Order: 96 78 54 53 32

Answer

```
#include <stdio.h>
#include <stdlib.h>

void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

int partition(int arr[], int low, int high, int iteration) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] >= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
```

```
        }
    }
    swap(&arr[i + 1], &arr[high]);

    printf("Iteration %d: ", iteration);
    for (int k = low; k <= high; k++) {
        printf("%d ", arr[k]);
    }
    printf("\n");

    return (i + 1);
}

void quickSort(int arr[], int low, int high, int* iteration_count) {
    if (low < high) {
        (*iteration_count)++;
        int pi = partition(arr, low, high, *iteration_count);
        quickSort(arr, low, pi - 1, iteration_count);
        quickSort(arr, pi + 1, high, iteration_count);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int* scores = (int*)malloc(n * sizeof(int));
    if (scores == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &scores[i]);
    }

    int iteration_count = 0;
    quickSort(scores, 0, n - 1, &iteration_count);

    printf("Sorted Order: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", scores[i]);
    }
    printf("\n");
```

```
    free(scores);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

You're a coach managing a list of finishing times for athletes in a race. The times are stored in an array, and you need to sort this array in ascending order to determine the rankings.

You'll use the insertion sort algorithm to accomplish this.

Input Format

The first line of input contains an integer n, representing the number of athletes.

The second line contains n space-separated integers, each representing the finishing time of an athlete in seconds.

Output Format

The output prints the sorted finishing times of the athletes in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
75 89 65 90 70
Output: 65 70 75 89 90

Answer

```
// You are using GCC
#include <stdio.h>

int main() {
    int n;
```

```
    scanf("%d", &n);
    int times[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &times[i]);
    }

    // Insertion sort ascending
    for (int i = 1; i < n; i++) {
        int key = times[i];
        int j = i - 1;
        while (j >= 0 && times[j] > key) {
            times[j + 1] = times[j];
            j--;
        }
        times[j + 1] = key;
    }

    for (int i = 0; i < n; i++) {
        printf("%d", times[i]);
        if (i != n - 1) printf(" ");
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Vishnu, a math enthusiast, is given a task to explore the magic of numbers. He has an array of positive integers, and his goal is to find the integer with the highest digit sum in the sorted array using the merge sort algorithm.

You have to assist Vishnu in implementing the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the array elements.

Output Format

The first line of output prints "The sorted array is: " followed by the sorted array, separated by a space.

The second line prints "The integer with the highest digit sum is: " followed by an integer representing the highest-digit sum.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
123 456 789 321 654
Output: The sorted array is: 123 321 456 654 789
The integer with the highest digit sum is: 789

Answer

```
// You are using GCC
#include <stdio.h>

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];

    for (int i = 0; i < n1; i++) L[i] = arr[left + i];
    for (int i = 0; i < n2; i++) R[i] = arr[mid + 1 + i];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) arr[k++] = L[i++];
        else arr[k++] = R[j++];
    }
    while (i < n1) arr[k++] = L[i++];
    while (j < n2) arr[k++] = R[j++];
}
```

```
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int digitSum(int num) {
    int sum = 0;
    while (num > 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}

int main() {
    int N;
    scanf("%d", &N);
    int arr[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    mergeSort(arr, 0, N - 1);

    printf("The sorted array is: ");
    for (int i = 0; i < N; i++) {
        printf("%d", arr[i]);
        if (i != N - 1) printf(" ");
    }

    int maxSum = digitSum(arr[0]);
    int maxNum = arr[0];
    for (int i = 1; i < N; i++) {
        int sum = digitSum(arr[i]);
        if (sum > maxSum) {
            maxSum = sum;
            maxNum = arr[i];
```

```
    }
  }

  printf("  The integer with the highest digit sum is: %d", maxNum);

  return 0;
}
```

Status : Correct **Marks : 10/10**

4. Problem Statement

Alex is working on a project that involves merging and sorting two arrays. He wants to write a program that merges two arrays, sorts the merged array in ascending order, removes duplicates, and prints the sorted array without duplicates.

Help Alex to implement the program using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of elements in the first array.

The second line consists of N integers, separated by spaces, representing the elements of the first array.

The third line consists of an integer M, representing the number of elements in the second array.

The fourth line consists of M integers, separated by spaces, representing the elements of the second array.

Output Format

The output prints space-separated integers, representing the merged and sorted array in ascending order, with duplicate elements removed.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
3
3 4 5

Output: 1 2 3 4 5

Answer

```
// You are using GCC
#include <stdio.h>

void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];

    for (int i = 0; i < n1; i++) L[i] = arr[left + i];
    for (int i = 0; i < n2; i++) R[i] = arr[mid + 1 + i];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) arr[k++] = L[i++];
        else arr[k++] = R[j++];
    }
    while (i < n1) arr[k++] = L[i++];
    while (j < n2) arr[k++] = R[j++];
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = (left + right) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

int main() {
    int N, M;
    scanf("%d", &N);
```

```
    int arr1[N];
    for (int i = 0; i < N; i++) scanf("%d", &arr1[i]);

    scanf("%d", &M);
    int arr2[M];
    for (int i = 0; i < M; i++) scanf("%d", &arr2[i]);

    int merged[N + M];
    for (int i = 0; i < N; i++) merged[i] = arr1[i];
    for (int i = 0; i < M; i++) merged[N + i] = arr2[i];

    mergeSort(merged, 0, N + M - 1);

    // Print merged array without duplicates
    printf("%d", merged[0]);
    for (int i = 1; i < N + M; i++) {
        if (merged[i] != merged[i - 1]) {
            printf(" %d", merged[i]);
        }
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**

5. Problem Statement

You are working on an optimization task for a sorting algorithm that uses insertion sort. Your goal is to determine the efficiency of the algorithm by counting the number of swaps needed to sort an array of integers.

Write a program that takes an array as input and calculates the number of swaps performed during the insertion sort process.

Example 1:

Input:

5

2 1 3 1 2

Output:

4

Explanation:

Step 1: [2, 1, 3, 1, 2] (No swaps)

Step 2: [1, 2, 3, 1, 2] (1 swap, element 1 shifts 1 place to the left)

Step 3: [1, 2, 3, 1, 2] (No swaps)

Step 4: [1, 1, 2, 3, 2] (2 swaps; element 1 shifts 2 places to the left)

Step 5: [1, 1, 2, 2, 3] (1 swap, element 2 shifts 1 place to the left)

Total number of swaps: 1 + 2 + 1 = 4

Example 2:

Input:

7

12 15 1 5 6 14 11

Output:

10

Explanation:

Step 1: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 2: [12, 15, 1, 5, 6, 14, 11] (1 swap, element 15 shifts 1 place to the left)

Step 3: [12, 15, 1, 5, 6, 14, 11] (No swaps)

Step 4: [1, 12, 15, 5, 6, 14, 11] (2 swaps, element 1 shifts 2 places to the left)

Step 5: [1, 5, 12, 15, 6, 14, 11] (1 swap, element 5 shifts 1 place to the left)

Step 6: [1, 5, 6, 12, 15, 14, 11] (2 swaps, element 6 shifts 2 places to the left)

Step 7: [1, 5, 6, 12, 14, 15, 11] (1 swap, element 14 shifts 1 place to the left)

Step 8: [1, 5, 6, 11, 12, 14, 15] (3 swaps, element 11 shifts 3 places to the left)

Total number of swaps: 1 + 2 + 1 + 2 + 1 + 3 = 10

Input Format

The first line of input consists of an integer n, representing the number of elements in the array.

The second line of input consists of n space-separated integers, representing the elements of the array.

Output Format

The output prints the number of swaps performed during the insertion sort process.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
2 1 3 1 2
Output: 4

Answer

```
// You are using GCC
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int swaps = 0;
    for(int i = 1; i < n; i++) {
```

```
        int key = arr[i];
        int j = i - 1;
        while(j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            swaps++;    // Count each shift as a swap
            j--;
        }
        arr[j+1] = key;
    }

    printf("%d", swaps);

    return 0;
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John and Mary are collaborating on a project that involves data analysis. They each have a set of age data, one sorted in ascending order and the other in descending order. However, their analysis requires the data to be in ascending order.

Write a program to help them merge the two sets of age data into a single sorted array in ascending order using merge sort.

Input Format

The first line of input consists of an integer N, representing the number of age values in each dataset.

The second line consists of N space-separated integers, representing the ages of participants in John's dataset (in ascending order).

The third line consists of N space-separated integers, representing the ages of participants in Mary's dataset (in descending order).

Output Format

The output prints a single line containing space-separated integers, which represents the merged dataset of ages sorted in ascending order.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 3 5 7 9
10 8 6 4 2

Output: 1 2 3 4 5 6 7 8 9 10

Answer

```
#include <stdio.h>

void merge(int arr[], int left[], int right[], int left_size, int right_size) {
    int i = 0, j = 0, k = 0;

    while (i < left_size && j < right_size) {
        if (left[i] < right[j]) {
            arr[k++] = left[i++];
        } else {
            arr[k++] = right[j++];
        }
    }

    while (i < left_size) {
        arr[k++] = left[i++];
    }

    while (j < right_size) {
        arr[k++] = right[j++];
    }
}
```

```
void mergeSort(int arr[], int size) {
    if (size < 2) return;

    int mid = size / 2;
    int left[mid];
    int right[size - mid];

    for (int i = 0; i < mid; i++) {
        left[i] = arr[i];
    }
    for (int i = mid; i < size; i++) {
        right[i - mid] = arr[i];
    }

    mergeSort(left, mid);
    mergeSort(right, size - mid);

    merge(arr, left, right, mid, size - mid);
}

int main() {
    int n, m;
    scanf("%d", &n);
    int arr1[n], arr2[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr1[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr2[i]);
    }
    int merged[n + n];
    mergeSort(arr1, n);
    mergeSort(arr2, n);
    merge(merged, arr1, arr2, n, n);
    for (int i = 0; i < n + n; i++) {
        printf("%d ", merged[i]);
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Nandhini asked her students to arrange a set of numbers in ascending order. She asked the students to arrange the elements using insertion sort, which involves taking each element and placing it in its appropriate position within the sorted portion of the array.

Assist them in the task.

Input Format

The first line of input consists of the value of n, representing the number of array elements.

The second line consists of n elements, separated by a space.

Output Format

The output prints the sorted array, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
67 28 92 37 59

Output: 28 37 59 67 92

Answer

```
#include <stdio.h>

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
```

```
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are the lead developer of a text-processing application that assists writers in organizing their thoughts. One crucial feature is a character-sorting service that helps users highlight the most critical elements of their text.

To achieve this, you decide to enhance the service to sort characters in descending order using the Quick-Sort algorithm. Implement the algorithm to efficiently rearrange the characters, ensuring that it is sorted in descending order.

Input Format

The first line of the input consists of a positive integer value N, representing the number of characters to be sorted.

The second line of input consists of N space-separated lowercase alphabetical characters.

Output Format

The output displays the set of alphabetical characters, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
a d g j k
Output: k j g d a

Answer

```
#include <stdio.h>
#include <string.h>

void swap(char* a, char* b) {
    char temp = *a;
    *a = *b;
    *b = temp;
}

int partition(char arr[], int low, int high) {
    char pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] > pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quicksort(char arr[], int low, int high) {
```

```
    if (low < high) {
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}

void printArray(char arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%c ", arr[i]);
    }
    printf("\n");
}
int main() {
    int n;
    scanf("%d", &n);

    char characters[n];

    for (int i = 0; i < n; i++) {
        char input;
        scanf(" %c", &input);
        characters[i] = input;
    }

    quicksort(characters, 0, n - 1);

    for (int i = 0; i < n; i++) {
        printf("%c ", characters[i]);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Kavya, a software developer, is analyzing data trends. She has a list of integers and wants to identify the nth largest number in the list after sorting the array using QuickSort.

To optimize performance, Kavya is required to use QuickSort to sort the list before finding the nth largest number.

Input Format

The first line of input consists of an integer n, representing the size of the array.

The second line consists of n space-separated integers, representing the elements of the array nums.

The third line consists of an integer k, representing the position of the largest

number you need to print after sorting the array.

Output Format

The output prints the k-th largest number in the sorted array (sorted in ascending order).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
-1 0 1 2 -1 -4
3
Output: 0

Answer

```
#include <stdio.h>
#include <stdlib.h>

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}

void quicksort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi - 1);
```

```
        quicksort(arr, pi + 1, high);
    }
}

void findNthLargest(int* nums, int n, int k) {
    quicksort(nums, 0, n - 1);
    printf("%d\n", nums[n - k]);
}

int main() {
    int n, k;
    scanf("%d", &n);
    int* nums = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    scanf("%d", &k);
    findNthLargest(nums, n, k);
    free(nums);
    return 0;
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Jose has an array of N fractional values, represented as double-point numbers. He needs to sort these fractions in increasing order and seeks your help.

Write a program to help Jose sort the array using the merge sort algorithm.

Input Format

The first line of input consists of an integer N, representing the number of fractions to be sorted.

The second line consists of N double-point numbers, separated by spaces, representing the fractions array.

Output Format

The output prints N double-point numbers, sorted in increasing order, and rounded to three decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
0.123 0.543 0.321 0.789

Output: 0.123 0.321 0.543 0.789

Answer

```
#include <stdio.h>
#include <stdlib.h>

int compare(double a, double b) {
    if (a < b) return -1;
    if (a > b) return 1;
    return 0;
}

void merge(double arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    double* L = (double*)malloc(n1 * sizeof(double));
    double* R = (double*)malloc(n2 * sizeof(double));

    for (int i = 0; i < n1; i++) {
        L[i] = arr[l + i];
    }
    for (int i = 0; i < n2; i++) {
        R[i] = arr[m + 1 + i];
    }

    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (compare(L[i], R[j]) <= 0) {
            arr[k] = L[i];
            i++;
```

```
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }

    free(L);
    free(R);
}

void mergeSort(double arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    double fractions[n];
    for (int i = 0; i < n; i++) {
        scanf("%lf", &fractions[i]);
    }
    mergeSort(fractions, 0, n - 1);
    for (int i = 0; i < n; i++) {
        printf("%.3f ", fractions[i]);
    }
```

```
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is building a basic hash table to manage student roll numbers for quick lookup. He decides to use Linear Probing to handle collisions.

Implement a hash table using linear probing where:

The hash function is: index = roll_number % table_sizeOn collision, check subsequent indexes (i+1, i+2, ...) until an empty slot is found.

You need to:

Insert a list of n student roll numbers into the hash table.Print the final state of the hash table. If a slot is empty, print -1.

Input Format

The first line of the input contains two integers n and table_size, where n is the

number of roll numbers to be inserted, and table_size is the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert into the hash table.

Output Format

The output should print a single line with table_size space-separated integers representing the final state of the hash table after all insertions.

If any slot remains unoccupied, it should be represented as -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4 7
50 700 76 85

Output: 700 50 85 -1 -1 -1 76

Answer

```
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1;
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size;
    while (table[index] != -1) {
        index = (index + 1) % size;
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
```

```
    for (int i = 0; i < n; i++) {
        int pos = linearProbe(table, size, arr[i]);
        table[pos] = arr[i];
    }
}

void printTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", table[i]);
    }
    printf("\n");
}

int main() {
    int n, table_size;
    scanf("%d %d", &n, &table_size);

    int arr[MAX];
    int table[MAX];

    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
    printTable(table, table_size);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Priya is developing a simple student management system. She wants to store roll numbers in a hash table using Linear Probing, and later search for specific roll numbers to check if they exist.

Implement a hash table using linear probing with the following operations:

Insert all roll numbers into the hash table.For a list of query roll numbers, print "Value x: Found" or "Value x: Not Found" depending on whether it exists in the table.

Input Format

The first line contains two integers, n and table_size — the number of roll numbers to insert and the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert.

The third line contains an integer q — the number of queries.

The fourth line contains q space-separated integers — the roll numbers to search for.

Output Format

The output print q lines — for each query value x, print: "Value x: Found" or "Value x: Not Found"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 10
21 31 41 51 61
3
31 60 51

Output: Value 31: Found
Value 60: Not Found
Value 51: Found

Answer

```
#include <stdio.h>

#define MAX 100

void initializeTable(int table[], int size) {
    for (int i = 0; i < size; i++) {
        table[i] = -1;
    }
}

int linearProbe(int table[], int size, int num) {
    int index = num % size;
    int i = 0;
    while (i < size) {
        int pos = (index + i) % size;
```

```
        if (table[pos] == -1 || table[pos] == -2) {
            return pos;
        }
        i++;
    }
    return -1;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for (int i = 0; i < n; i++) {
        int num = arr[i];
        int pos = linearProbe(table, size, num);
        if (pos != -1) {
            table[pos] = num;
        }
    }
}

int searchInHashTable(int table[], int size, int num) {
    int index = num % size;
    int i = 0;
    while (i < size) {
        int pos = (index + i) % size;
        if (table[pos] == -1) {
            return 0;
        }
        if (table[pos] == num) {
            return 1;
        }
        i++;
    }
    return 0;
}

int main() {
    int n, table_size;
    scanf("%d %d", &n, &table_size);

    int arr[MAX], table[MAX];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
```

```
    insertIntoHashTable(table, table_size, arr, n);

    int q, x;
    scanf("%d", &q);
    for (int i = 0; i < q; i++) {
        scanf("%d", &x);
        if (searchInHashTable(table, table_size, x))
            printf("Value %d: Found\n", x);
        else
            printf("Value %d: Not Found\n", x);
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a messaging application, users maintain a contact list with names and corresponding phone numbers. Develop a program to manage this contact list using a dictionary implemented with hashing.

The program allows users to add contacts, delete contacts, and check if a specific contact exists. Additionally, it provides an option to print the contact list in the order of insertion.

Input Format

The first line consists of an integer n, representing the number of contact pairs to be inserted.

Each of the next n lines consists of two strings separated by a space: the name of the contact (key) and the corresponding phone number (value).

The last line contains a string k, representing the contact to be checked or removed.

Output Format

If the given contact exists in the dictionary:

1. The first line prints "The given key is removed!" after removing it.
2. The next n - 1 lines print the updated contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

If the given contact does not exist in the dictionary:

1. The first line prints "The given key is not found!".
2. The next n lines print the original contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
Alice 1234567890
Bob 9876543210
Charlie 4567890123
Bob

Output: The given key is removed!
Key: Alice; Value: 1234567890
Key: Charlie; Value: 4567890123

Answer

```
void insertKeyValuePair(Dictionary *dict, const char *key, const char *value) {
   if (dict->size >= dict->capacity) {
      dict->capacity *= 2;
      dict->pairs = (KeyValuePair *)realloc(dict->pairs, dict->capacity *
sizeof(KeyValuePair));
   }
```

```
    strcpy(dict->pairs[dict->size].key, key);
    strcpy(dict->pairs[dict->size].value, value);
    dict->size++;
}

void removeKeyValuePair(Dictionary *dict, const char *key) {
    for (int i = 0; i < dict->size; i++) {
        if (strcmp(dict->pairs[i].key, key) == 0) {
            // Shift remaining elements
            for (int j = i; j < dict->size - 1; j++) {
                dict->pairs[j] = dict->pairs[j + 1];
            }
            dict->size--;
            return;
        }
    }
}

int doesKeyExist(Dictionary *dict, const char *key) {
    for (int i = 0; i < dict->size; i++) {
        if (strcmp(dict->pairs[i].key, key) == 0) {
            return 1;
        }
    }
    return 0;
}

void printDictionary(Dictionary *dict) {
    for (int i = 0; i < dict->size; i++) {
        printf("Key: %s; Value: %s\n", dict->pairs[i].key, dict->pairs[i].value);
    }
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Develop a program using hashing to manage a fruit contest where each fruit is assigned a unique name and a corresponding score. The program should allow the organizer to input the number of fruits and their names with scores.

Then, it should enable them to check if a specific fruit, identified by its name, is part of the contest. If the fruit is registered, the program should display its score; otherwise, it should indicate that it is not included in the contest.

Input Format

The first line consists of an integer N, representing the number of fruits in the contest.

The following N lines contain a string K and an integer V, separated by a space, representing the name and score of each fruit in the contest.

The last line consists of a string T, representing the name of the fruit to search for.

Output Format

If T exists in the dictionary, print "Key "T" exists in the dictionary.".

If T does not exist in the dictionary, print "Key "T" does not exist in the dictionary.".

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 2
banana 2
apple 1
Banana
Output: Key "Banana" does not exist in the dictionary.

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TABLE_SIZE 15

typedef struct Node {
    char key[21];        // Fruit name (max 20 characters + null terminator)
    int value;           // Score
    struct Node* next;
} Node;

Node* hashTable[TABLE_SIZE];

// Hash function to map a string to an index
unsigned int hash(char* key) {
```

```
    unsigned int hashValue = 0;
    while (*key) {
        hashValue = (hashValue * 31 + *key++) % TABLE_SIZE;
    }
    return hashValue;
}

// Insert a fruit into the hash table
void insert(char* key, int value) {
    unsigned int index = hash(key);
    Node* newNode = (Node*)malloc(sizeof(Node));
    strncpy(newNode->key, key, 20);
    newNode->value = value;
    newNode->next = hashTable[index];
    hashTable[index] = newNode;
}

// Search for a fruit in the hash table
Node* search(char* key) {
    unsigned int index = hash(key);
    Node* current = hashTable[index];
    while (current) {
        if (strcmp(current->key, key) == 0) {
            return current;
        }
        current = current->next;
    }
    return NULL;
}

int main() {
    int n;
    scanf("%d", &n);
    getchar(); // Consume newline after integer input

    // Initialize the hash table
    for (int i = 0; i < TABLE_SIZE; i++) {
        hashTable[i] = NULL;
    }

    // Insert fruits into the hash table
    for (int i = 0; i < n; i++) {
```

```
        char key[21];
        int value;
        scanf("%s %d", key, &value);
        insert(key, value);
    }

    // Read the fruit to be checked
    char t[21];
    scanf("%s", t);

    // Check if the fruit exists
    Node* fruit = search(t);
    if (fruit) {
        printf("Key \"%s\" exists in the dictionary.\n", t);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.\n", t);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Sivapoorani N
Email: 241001250@rajalakshmi.edu.in
Roll no: 241001250
Phone: 9363986514
Branch: REC
Department: I IT FC
Batch: 2028
Degree: B.E - IT

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are provided with a collection of numbers, each represented by an array of integers. However, there's a unique scenario: within this array, one element occurs an odd number of times, while all other elements occur an even number of times. Your objective is to identify and return the element that occurs an odd number of times in this arrangement.

Utilize mid-square hashing by squaring elements and extracting middle digits for hash codes. Implement a hash table for efficient integer occurrence tracking.

Note: Hash function: squared = key * key.

Example

Input:

7

2 2 3 3 4 4 5

Output:

5

Explanation

The hash function and the calculated hash indices for each element are as follows:

2 -> hash(2*2) % 100 = 4

3 -> hash(3*3) % 100 = 9

4 -> hash(4*4) % 100 = 16

5 -> hash(5*5) % 100 = 25

The hash table records the occurrence of each element's hash index:

Index 4: 2 occurrences

Index 9: 2 occurrences

Index 16: 2 occurrences

Index 25: 1 occurrence

Among the elements, the integer 5 occurs an odd number of times (1 occurrence) and satisfies the condition of the problem. Therefore, the program outputs 5.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer representing the element that occurs an odd

number of times.

If no such element exists, print -1.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
2 2 3 3 4 4 5

Output: 5

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX_SIZE 100

unsigned int hash(int key, int tableSize) {
    long long squared = (long long)key * key;
    return (unsigned int)(squared % tableSize);
}

int getOddOccurrence(int arr[], int size) {
    unsigned int hashCounts[MAX_SIZE] = {0};

    for (int i = 0; i < size; i++) {
        unsigned int index = hash(arr[i], MAX_SIZE);
        hashCounts[index]++;
    }

    for (int i = 0; i < size; i++) {
        unsigned int index = hash(arr[i], MAX_SIZE);
        if (hashCounts[index] % 2 == 1) {
            return arr[i];
        }
    }
```

```
    return -1;
}
int main() {
    int n;
    scanf("%d", &n);

    int arr[MAX_SIZE];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("%d\n", getOddOccurrence(arr, n));

    return 0;
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Sunmathi S
Email: 241801283@rajalakshmi.edu.in
Roll no: 241801283
Phone: 9940554830
Branch: REC
Department: I AI & DS FD
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a messaging application, users maintain a contact list with names and corresponding phone numbers. Develop a program to manage this contact list using a dictionary implemented with hashing.

The program allows users to add contacts, delete contacts, and check if a specific contact exists. Additionally, it provides an option to print the contact list in the order of insertion.

Input Format

The first line consists of an integer n, representing the number of contact pairs to be inserted.

Each of the next n lines consists of two strings separated by a space: the name of the contact (key) and the corresponding phone number (value).

The last line contains a string k, representing the contact to be checked or removed.

Output Format

If the given contact exists in the dictionary:

1. The first line prints "The given key is removed!" after removing it.
2. The next n - 1 lines print the updated contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

If the given contact does not exist in the dictionary:

1. The first line prints "The given key is not found!".
2. The next n lines print the original contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
Alice 1234567890
Bob 9876543210
Charlie 4567890123
Bob

Output: The given key is removed!
Key: Alice; Value: 1234567890
Key: Charlie; Value: 4567890123

Answer

```
void insertKeyValuePair(Dictionary *dict, const char *key, const char *value) {
   if (dict->size >= dict->capacity) {
      dict->capacity *= 2;
      dict->pairs = (KeyValuePair *)realloc(dict->pairs, dict->capacity *
sizeof(KeyValuePair));
```

```
    }
    strcpy(dict->pairs[dict->size].key, key);
    strcpy(dict->pairs[dict->size].value, value);
    dict->size++;
}

void removeKeyValuePair(Dictionary *dict, const char *key) {
    for (int i = 0; i < dict->size; i++) {
        if (strcmp(dict->pairs[i].key, key) == 0) {
            for (int j = i; j < dict->size - 1; j++) {
                dict->pairs[j] = dict->pairs[j + 1];
            }
            dict->size--;
            return;
        }
    }
}

int doesKeyExist(Dictionary *dict, const char *key) {
    for (int i = 0; i < dict->size; i++) {
        if (strcmp(dict->pairs[i].key, key) == 0) {
            return 1;
        }
    }
    return 0;
}

void printDictionary(Dictionary *dict) {
    for (int i = 0; i < dict->size; i++) {
        printf("Key: %s; Value: %s\n", dict->pairs[i].key, dict->pairs[i].value);
    }
}
```

Status : Correct **Marks : 10/10**