Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

Input Format

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

The second line consists of N space-separated elements of the linked list.

The third line consists of an integer x, representing the position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Amar is working on a project where he needs to implement a special type of queue that allows selective dequeuing based on a given multiple. He wants to efficiently manage a queue of integers such that only elements not divisible by a given multiple are retained in the queue after a selective dequeue operation.

Implement a program to assist Amar in managing his selective queue.

Example

Input:

5

10 2 30 4 50

5

Output:

Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Explanation:

After selective dequeue with a multiple of 5, the elements that are multiples of 5 should be removed. Therefore, only 10, 30, and 50 should be removed from the queue. The updated Queue is 2 4.

Input Format

The first line contains an integer n, representing the number of elements initially present in the queue.

The second line contains n space-separated integers, representing the elements of the queue.

The third line contains an integer multiple, representing the divisor for selective dequeue operation.

Output Format

The first line of output prints "Original Queue: " followed by the space-separated elements in the queue before the dequeue operation.

The second line prints "Queue after selective dequeue: " followed by the remaining space-separated elements in the queue, after deleting elements that are the multiples of the specified number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
10 2 30 4 50
5
Output: Original Queue: 10 2 30 4 50
Queue after selective dequeue: 2 4

Answer

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

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

    printf("Original Queue: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", queue[i]);
    }
    printf("\nQueue after selective dequeue: ");
    for (int i = 0; i < n; i++) {
        if (queue[i] % multiple != 0) {
            printf("%d ", queue[i]);
        }
    }
    return 0;
}
```

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

2. Problem Statement

You are tasked with developing a simple ticket management system for a customer support department. In this system, customers submit support tickets, which are processed in a First-In-First-Out (FIFO) order. The system needs to handle the following operations:

Ticket Submission (Enqueue Operation): New tickets are submitted by customers. Each ticket is assigned a unique identifier (represented by an integer). When a new ticket arrives, it should be added to the end of the queue.

Ticket Processing (Dequeue Operation): The support team processes tickets in the order they are received. The ticket at the front of the queue is processed first. After processing, the ticket is removed from the queue.

Display Ticket Queue: The system should be able to display the current state of the ticket queue, showing the sequence of ticket identifiers from front to rear.

Input Format

The first input line contains an integer n, the number of tickets submitted by customers.

The second line consists of a single integer, representing the unique identifier of each submitted ticket, separated by a space.

Output Format

The first line displays the "Queue: " followed by the ticket identifiers in the queue after all tickets have been submitted.

The second line displays the "Queue After Dequeue: " followed by the ticket identifiers in the queue after processing (removing) the ticket at the front.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 6
14 52 63 95 68 49

Output: Queue: 14 52 63 95 68 49
Queue After Dequeue: 52 63 95 68 49

Answer

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

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

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

    printf("Queue: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", queue[i]);
    }
    printf("\nQueue After Dequeue: ");
    for (int i = 1; i < n; i++) {
        printf("%d ", queue[i]);
    }
    return 0;
}
```

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

3. Problem Statement

You've been assigned the challenge of developing a queue data structure using a linked list.

The program should allow users to interact with the queue by enqueuing positive integers and subsequently dequeuing and displaying elements.

Input Format

The input consists of a series of integers, one per line. Enter positive integers into the queue.

Enter -1 to terminate input.

Output Format

The output prints the space-separated dequeued elements.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1
2
3
4
-1

Output: Dequeued elements: 1 2 3 4

Answer

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

struct Node {
   int data;
   struct Node* next;
};

struct Queue {
   struct Node* front;
   struct Node* rear;
};

void enqueue(struct Queue* q, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = NULL;
   if (q->rear == NULL) {
      q->front = q->rear = newNode;
      return;
   }
   q->rear->next = newNode;
   q->rear = newNode;
}

int dequeue(struct Queue* q) {
   if (q->front == NULL) {
      printf("Queue Underflow\n");
      return -1;
   }
   struct Node* temp = q->front;
```

```
    int value = temp->data;
    q->front = q->front->next;
    if (q->front == NULL) {
        q->rear = NULL;
    }
    free(temp);
    return value;
}

int main() {
    struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
    q->front = q->rear = NULL;
    int value;

    while (1) {
        scanf("%d", &value);
        if (value == -1) {
            break;
        }
        if (value > 0) {
            enqueue(q, value);
        } else {
            printf("Please enter a positive integer.\n");
        }
    }
    printf("Dequeued elements: ");
    while (q->front != NULL) {
        printf("%d ", dequeue(q));
    }
    printf("\n");
    free(q);
    return 0;
}
```

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

4. Problem Statement

Sharon is developing a queue using an array. She wants to provide the functionality to find the Kth largest element. The queue should support the addition and retrieval of the Kth largest element effectively. The maximum capacity of the queue is 10.

Assist her in the program.

Input Format

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

The second line consists of N space-separated integers.

The third line consists of an integer K.

Output Format

For each enqueued element, print a message: "Enqueued: " followed by the element.

The last line prints "The [K]th largest element: " followed by the Kth largest element.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 45 93 87 25
4
Output: Enqueued: 23
Enqueued: 45
Enqueued: 93
Enqueued: 87
Enqueued: 25
The 4th largest element: 25

Answer

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

int compare(const void *a, const void *b) {
    return (*(int *)b - *(int *)a);
}

int main() {
    int queue[10];
    int n, k;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
        printf("Enqueued: %d\n", queue[i]);
    }

    scanf("%d", &k);
    qsort(queue, n, sizeof(int), compare);
    printf("The %dth largest element: %d\n", k, queue[k - 1]);

    return 0;
}
```

Status : Correct **_Marks : 10/10_**

5. Problem Statement

Guide Harish in developing a simple queue system for a customer service center. The customer service center can handle up to 25 customers at a time. The queue needs to support basic operations such as adding a customer to the queue, serving a customer (removing them from the queue), and displaying the current queue of customers.

Use an array for implementation.

Input Format

The first line of the input consists of an integer N, the number of customers arriving at the service center.

The second line consists of N space-separated integers, representing the customer IDs in the order they arrive.

Output Format

After serving the first customer in the queue, display the remaining customers in the queue.

If a dequeue operation is attempted on an empty queue, display "Underflow".

If the queue is empty, display "Queue is empty".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 102 103 104 105
Output: 102 103 104 105

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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 index = linearProbe(table, size, arr[i]);
        table[index] = arr[i];
    }
}

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



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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

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

```
#include <stdio.h>

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

   int L[20], R[20];

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

   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 - left) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

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

int main() {
    int n, m, i, j;
    int a[10], b[10], merged[20], result[20];

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

    scanf("%d", &m);
    for (i = 0; i < m; i++)
        scanf("%d", &b[i]);

    for (i = 0; i < n; i++)
        merged[i] = a[i];
    for (j = 0; j < m; j++)
        merged[n + j] = b[j];

    int total = n + m;
    mergeSort(merged, 0, total - 1);
    int idx = 0;
    result[idx++] = merged[0];
    for (i = 1; i < total; i++) {
        if (merged[i] != merged[i - 1])
            result[idx++] = merged[i];
    }

    for (i = 0; i < idx; i++)
        printf("%d ", result[i]);

    printf("\n");
    return 0;
}
```

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

2. 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, i, j, key, swaps = 0;
    int arr[10];
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
            swaps++;
        }
        arr[j + 1] = key;
    }

    printf("%d\n", swaps);
    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 l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[10], R[10];

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

    i = 0;
    j = 0;
    k = l;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            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++;
    }
}

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

int main() {
    int n, i;
    int arr[10];
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    mergeSort(arr, 0, n - 1);

    printf("The sorted array is: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");

    int maxSum = 0, maxNum = arr[0];
    for (i = 0; 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\n", maxNum);
    return 0;
}
```

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

4. 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, i, j, key;
   scanf("%d", &n);
   int arr[20];

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

```
    }

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

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

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

    return 0;
}
```

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

5. 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>

#define MAX 10

int iteration = 1;

// Function to swap two elements
void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}

// Partition function for descending order
int partition(int arr[], int low, int high) {
   int pivot = arr[high]; // Choosing the last element as pivot
   int i = low - 1;

   for (int j = low; j < high; j++) {
        // For descending order, use '>=' comparison
        if (arr[j] >= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

// Quick Sort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

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

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int n;
    int arr[MAX];

    // Read the number of scores
    scanf("%d", &n);

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

    // Perform Quick Sort
    quickSort(arr, 0, n - 1);
    // Display the sorted scores
    printf("Sorted Order:");
    for (int i = 0; i < n; i++) {
        printf(" %d", arr[i]);
    }
    printf("\n");

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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;
    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 index = linearProbe(table, size, arr[i]);
        table[index] = arr[i];
    }
}

int searchInHashTable(int table[], int size, int num) {
    int index = num % size;
    int start = index;

    while (table[index] != -1) {
        if (table[index] == num)
            return 1;
        index = (index + 1) % size;
        if (index == start)
            break;
    }
    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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 0_Arrays and Functions

Attempt : 1
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. Problem Statement

Tim is creating a program to track and analyze student attendance. The program requires two inputs: the total number of students (n) and the total number of class sessions (m). The task is to design and populate an attendance matrix, 'matrix', representing the attendance record of each student for each session.

The program's specific objective is to determine whether the last student on the list attended an even or odd number of classes. This functionality will aid teachers in quickly evaluating the attendance habits of individual students.

Input Format

The first line of input consists of a positive integer n, representing the number of students.

The second line consists of a positive integer m, representing the number of class sessions.

The next n lines consist of m space-separated positive integers representing the number of classes attended by the student.

Output Format

The output displays one of the following results:

If the last session is even the output prints "[LastSession] is even".

If the last session is odd the output prints "[LastSession] is odd".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
2
1 2
3 100
Output: 100 is even

Answer

```
// You are using GCC
#include<stdio.h>
int main()
{
   int n, m;
   scanf("%d %d", &n, &m);
   int matrix[n][m];
   for(int i = 0; i < n; i++)
   {
      for(int j = 0; j < m; j++)
      {
         scanf("%d", &matrix[i][j]);
      }
   }
   if(matrix[n-1][m-1] % 2 == 0)
```

```
    {
        printf("%d is even", matrix[n-1][m-1]);
    }
    else
    {
        printf("%d is odd", matrix[n-1][m-1]);
    }
}
```

Status : Correct ***Marks : 1/1***

2. Problem Statement

Write a program that will read a Matrix (two-dimensional arrays) and print the sum of all elements of each row by passing the matrix to a function.

Function Signature: void calculateRowSum(int [ ][ ], int, int)

Input Format

The first line consists of an integer M representing the number of rows.

The second line consists of an integer N representing the number of columns.

The next M lines consist of N space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the sum of all elements of each row separated by a space.

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Output: 6 15 24

Answer

```
#include <stdio.h>

// You are using GCC
void calculateRowSum(int matrix[20][20], int rows, int cols) {
   for(int i = 0; i < rows; i++){
      int sum = 0;
      for(int j = 0; j < cols; j++){
         sum += matrix[i] [j];
      }
printf("%d ", sum);
   }
}

int main() {
   int matrix[20][20];
   int r, c;

   scanf("%d", &r);
   scanf("%d", &c);

   for (int i = 0; i < r; i++) {
      for (int j = 0; j < c; j++) {
         scanf("%d", &matrix[i][j]);
      }
   }

   calculateRowSum(matrix, r, c);
   return 0;
}
```

Status : Correct ***Marks : 1/1***

3. Problem Statement

Write a program that reads an integer 'n' and a square matrix of size 'n x n' from the user. The program should then set all the elements in the lower triangular part of the matrix (including the main diagonal) to zero using a function and display the resulting matrix.

Function Signature: void setZeros(int [ ][ ], int)

Input Format

The first line consists of an integer M representing the number of rows & columns.

The next M lines consist of M space-separated integers in each line representing the elements of the matrix.

Output Format

The output displays the matrix containing M space-separated elements in M lines where the lower triangular elements are replaced with zero.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
10 20 30
40 50 60
70 80 90

Output: 0 20 30
0 0 60
0 0 0

Answer

```
#include <stdio.h>

// You are using GCC
void setZeros(int arr[10][10], int n) {
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            if(!(j > i)) {
                arr[i][j] = 0;
            }
        }
        printf("\n");
    }
    //Type your code here
}
int main() {
   int arr1[10][10];
   int n;

   scanf("%d", &n);

   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         scanf("%d", &arr1[i][j]);
      }
   }

   setZeros(arr1, n);

   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         printf("%d ", arr1[i][j]);
      }
      printf("\n");
   }

   return 0;
}
```

Status : Correct ***Marks : 1/1***

4. Problem Statement

Alex, a budding programmer, is tasked with writing a menu-driven program to perform operations on an array of integers. The operations include finding the smallest number, the largest number, the sum of all numbers, and their average. The program must repeatedly display the menu until Alex chooses to exit.

Write a program to ensure the specified tasks are implemented based on Alex's choices.

Input Format

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

The second line contains n space-separated integers representing the array elements.

The subsequent lines contain integers representing the menu choices:

Choice 1: Find and display the smallest number.

Choice 2: Find and display the largest number.

Choice 3: Calculate and display the sum of all numbers.

Choice 4: Calculate and display the average of all numbers as double.

Choice 5: Exit the program.

Output Format

For each valid menu choice, print the corresponding result:

For choice 1, print "The smallest number is: X", where X is the smallest number in the array.

For choice 2, print "The largest number is: X", where X is the largest number in the array.

For choice 3, print "The sum of the numbers is: X", where X is the sum of all numbers in the array.

For choice 4, print "The average of the numbers is: X. XX", where X.XX is the double value representing an average of all numbers in the array, rounded to two decimal places.

For choice 5, print "Exiting the program".

If an invalid choice is made, print "Invalid choice! Please enter a valid option (1-5)."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
10 20 30
1
5

Output: The smallest number is: 10
Exiting the program

Answer

```
// You are using GCC
#include<stdio.h>
void FindSmall(int arr[],int n)
{
   int min=arr[0];
   for(int i=0;i<n;i++)
   {
      if(arr[i]<min)
      {
         min=arr[i];
      }
   }
   printf("The smallest number is:%d\n",min);
}
void FindLarge(int arr[],int n)
{
   int max=arr[0];
   for(int i=0;i<n;i++)
   {
      if(arr[i]>max)
      {
         max=arr[i];
      }
   }
   printf("The largest number is:%d\n",max);
}
void FindSum(int arr[],int n)
{
 int sum=0;
 for(int i=0;i<n;i++)
 {
     sum+=arr[i];
  }
  printf("The sum of the numbers is:%d\n",sum);
}
void FindAvg(int arr[],int n)
{
   int sum=0;
   for(int i=0;i<n;i++)
   {
      sum+=arr[i];
   }
   float avg=(float)sum/n;
   printf("the average of the numbers is:%.2f\n",avg);
}
int main()
{
   int n,choice;
   scanf("%d",&n);
   int arr[n];
   for(int i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
   scan:
   scanf("%d",&choice);
   switch(choice)
   {
      case 1:
      FindSmall(arr,n);
      goto scan;
      break;
      case 2:
      FindLarge(arr,n);
      goto scan;
      break;
      case 3:
      FindSum(arr,n);
      goto scan;
      break;
      case 4:
      FindAvg(arr,n);
      goto scan;
```

```
        break;
        default:
        printf("Invalidchoice!Please enter a valid option(1-5).\n");
        case 5:
        printf("Exiting the program\n");
        break;
    }
    return 0;
}
```

Status : Correct ***Marks : 1/1***

5. Problem Statement

Saurabh is the manager of a growing tech company. He needs a program to record and analyze the monthly salaries of his employees. The program will take the number of employees and their respective salaries as input and then calculate the average salary, and find the highest and lowest salary among them.

Help Saurabh automate this task efficiently.

Input Format

The first line of input consists of an integer n, representing the number of employees.

The second line consists of n integers, where each integer represents the salary of an employee.

Output Format

The output prints n lines, where each line will display: "Employee i: "Salary

Where i is the employee number (starting from 1) and salary is the respective salary of that employee.

After that, print the average salary in the following format: "Average Salary: "average_salary

Where average_salary is the average salary of all employees, rounded to two decimal places.

Next, print the highest salary in the following format: "Highest Salary: "max_salary

Where max_salary is the highest salary among all employees.

Finally, print the lowest salary in the following format:"Lowest Salary: "min_salary

Where min_salary is the lowest salary among all employees.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
4000
3500
6000
2500
4500

Output: Employee 1: 4000
Employee 2: 3500
Employee 3: 6000
Employee 4: 2500
Employee 5: 4500

Average Salary: 4100.00
Highest Salary: 6000
Lowest Salary: 2500

Answer

```
// You are using GCC
#include<stdio.h>
int main()
{
   int n, avg = 0, high = 0, low = 0;
   scanf("%d", &n);
   int arr[n];
   for(int i = 0; i < n; i++)
   {
      scanf("%d",&arr[i]);
      avg += arr[i];
      if(high < arr[i])
      {
         high = arr[i];
      }
      if(i == 0)
      {
       low = arr[i];
      }
      else if(low > arr[i])
      {
         low = arr[i];
      }
   }
   for(int i = 0; i < n; i++)
   {
      printf("Employee %d: %d\n", i+1, arr[i]);
   }

printf("\n");
printf("Average Salary: %.2f\n", (float)avg / (float)n);
printf("Highest Salary: %d\n", high);
printf("Lowest Salary: %d", low);
}
```

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Siri is a computer science student who loves solving mathematical problems. She recently learned about infix and postfix expressions and was fascinated by how they can be used to evaluate mathematical expressions.

She decided to write a program to convert an infix expression with operators to its postfix form. Help Siri in writing the program.

Input Format

The input consists of a single line containing an infix expression.

Output Format

The output prints a single line containing the postfix expression equivalent to the given infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: (2 + 3) * 4
Output: 23+4*

Answer

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

#define MAX 50


char stack[MAX];
int top = -1;


int isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/' || c == '(' || c == ')');
}


int precedence(char c) {
    if (c == '+' || c == '-') {
        return 1;
    } else if (c == '*' || c == '/') {
        return 2;
    } else {
        return 0;
    }
}


void push(char c) {
    if (top == MAX - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack[++top] = c;
}


char pop() {
    if (top == -1) {
        return -1;
    }
    return stack[top--];
}


void infixToPostfix(char* infix) {
    int i = 0, k = 0;
    char postfix[MAX];

    while (infix[i]) {

        if (infix[i] == ' ') {
            i++;
            continue;
        }


        if (isdigit(infix[i])) {
            postfix[k++] = infix[i];
        }

        else if (infix[i] == '(') {
            push(infix[i]);
        }

        else if (infix[i] == ')') {
            while (top != -1 && stack[top] != '(') {
                postfix[k++] = pop();
            }
            pop();
        }
        else if (isOperator(infix[i])) {
            while (top != -1 && precedence(stack[top]) >= precedence(infix[i])) {
                postfix[k++] = pop();
            }
            push(infix[i]);
        }
        i++;
    }


    while (top != -1) {
        postfix[k++] = pop();
    }

    postfix[k] = '\0';
    printf("%s\n", postfix);
}

int main() {
    char infix[MAX];


    fgets(infix, MAX, stdin);


    infix[strcspn(infix, "\n")] = '\0';


    infixToPostfix(infix);

    return 0;
}
```

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

2. Problem Statement

Latha is taking a computer science course and has recently learned about infix and postfix expressions. She is fascinated by the idea of converting infix expressions into postfix notation. To practice this concept, she wants to implement a program that can perform the conversion for her.

Help Latha by designing a program that takes an infix expression as input and outputs its equivalent postfix notation.

Example

Input:

(3+4)5

Output:

34+5

Input Format

The input consists of a string, the infix expression to be converted to postfix notation.

Output Format

The output displays a string, the postfix expression equivalent of the input infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: A+B*C-D/E
Output: ABC*+DE/-

Answer

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

#define MAX 100

char stack[MAX];
int top = -1;
```

```
void push(char c) {
    if (top == MAX - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack[++top] = c;
}

char pop() {
    if (top == -1) {
        return -1;
    }
    return stack[top--];
}

int precedence(char c) {
    if (c == '+' || c == '-') {
        return 1;
    } else if (c == '*' || c == '/') {
        return 2;
    } else if (c == '^') {
        return 3;
    }
    return 0;
}

void infixToPostfix(char* infix, char* postfix) {
    int i = 0, k = 0;
    char symbol;
    while ((symbol = infix[i++]) != '\0') {
        if (isalnum(symbol)) {
            postfix[k++] = symbol;
        } else if (symbol == '(') {
            push(symbol);
        } else if (symbol == ')') {
            while (top != -1 && stack[top] != '(') {
                postfix[k++] = pop();
            }
            pop();
        } else {
            while (top != -1 && precedence(stack[top]) >= precedence(symbol)) {
```

```
            postfix[k++] = pop();
        }
        push(symbol);
      }
    }
    while (top != -1) {
      postfix[k++] = pop();
    }
    postfix[k] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];
    fgets(infix, MAX, stdin);
    infixToPostfix(infix, postfix);
    printf("%s", postfix);
    return 0;
}
```

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

3. Problem Statement

Raj is a software developer, and his team is building an application that processes user inputs in the form of strings containing brackets. One of the essential features of the application is to validate whether the input string meets specific criteria.

During testing, Raj inputs the string "(([]))){}". The application correctly returns "Valid string" because the input satisfies the criteria: every opening bracket (, [, and { has a corresponding closing bracket),], and }, arranged in the correct order.

Next, Raj tests the application with the string "([)]". This time, the application correctly returns "Invalid string" because the opening bracket [is incorrectly closed by the bracket), which violates the validation rules.

Finally, Raj enters the string "{[()]}". The application correctly identifies it as a "Valid string" since all opening brackets are matched with the corresponding closing brackets in the correct order.

As a software developer, Raj's responsibility is to ensure that the application works reliably and produces accurate results for all input strings, following the validation rules. He accomplishes this by using a method for solving such problems.

Input Format

The input comprises a string representing a sequence of brackets that need to be validated.

Output Format

The output prints "Valid string" if the string is valid. Otherwise, it prints "Invalid string".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: (([])){}
Output: Valid string

Answer

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

#define MAX 100

char stack[MAX];
int top = -1;

void push(char c) {
   if (top == MAX - 1) {
      printf("Stack Overflow\n");
      return;
   }
   stack[++top] = c;
}
```

```
char pop() {
    if (top == -1) {
        return -1;
    }
    return stack[top--];
}

int isMatchingPair(char open, char close) {
    if (open == '(' && close == ')') return 1;
    if (open == '{' && close == '}') return 1;
    if (open == '[' && close == ']') return 1;
    return 0;
}

int isValidString(char* exp) {
    int i = 0;
    char symbol;
    while ((symbol = exp[i++]) != '\0') {
        if (symbol == '(' || symbol == '[' || symbol == '{') {
            push(symbol);
        } else if (symbol == ')' || symbol == ']' || symbol == '}') {
            if (top == -1) {
                return 0;
            }
            char open = pop();
            if (!isMatchingPair(open, symbol)) {
                return 0;
            }
        }
    }
    return top == -1;
}

int main() {
    char exp[MAX];
    fgets(exp, MAX, stdin);
    if (isValidString(exp)) {
        printf("Valid string\n");
    } else {
        printf("Invalid string\n");
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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***

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Write a program to implement a queue using an array and pointers. The program should provide the following functionalities:

Insert an element into the queue.Delete an element from the queue.Display the elements in the queue.

The queue has a maximum capacity of 5 elements. If the queue is full and an insertion is attempted, a "Queue is full" message should be displayed. If the queue is empty and a deletion is attempted, a "Queue is empty" message should be displayed.

Input Format

Each line contains an integer representing the chosen option from 1 to 3.

Option 1: Insert an element into the queue followed by an integer representing the element to be inserted, separated by a space.

Option 2: Delete an element from the queue.

Option 3: Display the elements in the queue.

Output Format

For option 1 (insertion):-

1. The program outputs: "<data> is inserted in the queue." if the data is successfully inserted.
2. "Queue is full." if the queue is already full and cannot accept more elements.

For option 2 (deletion):-

1. The program outputs: "Deleted number is: <data>" if an element is successfully deleted and returns the value of the deleted element.
2. "Queue is empty." if the queue is empty no elements can be deleted.

For option 3 (display):-

1. The program outputs: "Elements in the queue are: <element1> <element2> ... <elementN>" where <element1>, <element2>, ..., <elementN> represent the elements present in the queue.
2. "Queue is empty." if the queue is empty no elements can be displayed.

For invalid options, the program outputs: "Invalid option."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 10

```
3
5
Output: 10 is inserted in the queue.
Elements in the queue are: 10
Invalid option.
```

Answer

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

#define max 5

int queue[max];
int front = -1, rear = -1;

int insertq(int *data) {
    if (rear == max - 1) {
        return 0;
    }
    if (front == -1) {
        front = 0;
    }
    rear++;
    queue[rear] = *data;
    return 1;
}

int delq() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return -1;
    }
    int val = queue[front];
    front++;
    if (front > rear) {
        front = rear = -1;
    }
    printf("Deleted number is: %d\n", val);
    return val;
}

void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Elements in the queue are:");
    for (int i = front; i <= rear; i++) {
        printf(" %d", queue[i]);
    }
    printf("\n");
}




int main()
{
    int data, reply, option;
    while (1)
    {
        if (scanf("%d", &option) != 1)
            break;
        switch (option)
        {
            case 1:
                if (scanf("%d", &data) != 1)
                    break;
                reply = insertq(&data);
                if (reply == 0)
                    printf("Queue is full.\n");
                else
                    printf("%d is inserted in the queue.\n", data);
                break;
            case 2:
                delq();  //    Called without arguments
                break;
            case 3:
                display();
                break;
            default:
                printf("Invalid option.\n");
                break;
        }
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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) {
   int squared = key * key;
   int numDigits = 0;
   int temp = squared;
   while (temp > 0) {
      temp /= 10;
      numDigits++;
   }
   int middleDigits = (numDigits / 2);
   int divisor = 1;
   for (int i = 0; i < middleDigits; i++) {
      divisor *= 10;
   }
   return (squared / divisor) % tableSize;
}

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

```
        hashTable[index]++;
    }
    for (int i = 0; i < size; i++) {
        int index = hash(arr[i], MAX_SIZE);
        if (hashTable[index] % 2 != 0) {
            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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 1

Attempt : 2
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>

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left = arr[:mid]
        right = arr[mid:]

        merge_sort(left)
        merge_sort(right)

        i = j = k = 0

        # Merge left and right halves
        while i < len(left) and j < len(right):
            if left[i] < right[j]:
                arr[k] = left[i]
                i += 1
            else:
                arr[k] = right[j]
                j += 1
            k += 1
        # Copy remaining elements
        while i < len(left):
            arr[k] = left[i]
            i += 1
            k += 1

        while j < len(right):
            arr[k] = right[j]
            j += 1
            k += 1

# Input reading
n = int(input())
john_data = list(map(int, input().split()))
mary_data = list(map(int, input().split()))

# Convert Mary's data to ascending order
mary_data.reverse()

# Merge the two datasets
combined_data = john_data + mary_data

# Sort using merge sort
merge_sort(combined_data)

# Output the result
print(' '.join(map(str, combined_data)))
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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The output prints the sum of the coefficients of the polynomials.

Sample Test Case

Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0

Output: 18

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct Polynomial
{
   int coefficient;
   int exponential;
   struct Polynomial*next;
}Node;
Node*newnode(int coefficient,int exponential)
{
   Node*new_node=(Node*)malloc(sizeof(Node));
   new_node->coefficient = coefficient;
   new_node->exponential = exponential;
   new_node->next = NULL;
   return new_node;
}
Node*input(int n)
{
   int c,e;
```

```
    scanf("%d %d",&c,&e);
    Node*poly=newnode(c,e);
    Node*ptr=poly;
    for(int i=1;i<n;i++)
    {
        scanf("%d %d",&c,&e);
        ptr->next = newnode(c,e);
        ptr = ptr->next;
    }
    return poly;

}
int csum(Node*poly)
{
    int sum=0;
    Node* ptr=poly;
    while(ptr)
    {
        sum += ptr->coefficient;
        ptr = ptr->next;
    }
    return sum;
}
int main()
{
    int sum=0;
    int n;
    scanf("%d",&n);
    Node*poly1 = input(n);
    scanf("%d",&n);
    Node*poly2 = input(n);
    sum += csum(poly1);
    sum += csum(poly2);
    printf("%d",sum);
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

```
int keyExists(KeyValuePair* dictionary, int size, const char* key) {
   for (int i = 0; i < size; i++) {
      if (strcmp(dictionary[i].key, key) == 0) {
         return 1;
      }
   }
   return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

Input Format

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

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

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8

Output: 8 4 6 10 12

Answer

-

Status : Skipped ***Marks : 0/10***

2. Problem Statement

Ashiq is developing a ticketing system for a small amusement park. The park issues tickets to visitors in the order they arrive. However, due to a system change, the oldest ticket (first inserted) must be revoked instead of the last one.

To manage this, Ashiq decided to use a doubly linked list-based stack, where:

Pushing adds a new ticket to the top of the stack.Removing the first inserted ticket (removing from the bottom of the stack).Printing the remaining tickets from bottom to top.

Input Format

The first line consists of an integer n, representing the number of tickets issued.

The second line consists of n space-separated integers, each representing a ticket number in the order they were issued.

Output Format

The output prints space-separated integers, representing the remaining ticket numbers in the order from bottom to top.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
24 96 41 85 97 91 13

Output: 96 41 85 97 91 13

Answer

-

Status : - ***Marks : 0/10***

3. Problem Statement

Krishna needs to create a doubly linked list to store and display a sequence of integers. Your task is to help write a program to read a list of integers from input, store them in a doubly linked list, and then display the list.

Input Format

The first line of input consists of an integer n, representing the number of integers in the list.

The second line of input consists of n space-separated integers.

Output Format

The output prints a single line displaying the integers in the order they were added to the doubly linked list, separated by spaces.

If nothing is added (i.e., the list is empty), it will display "List is empty".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

Output: 1 2 3 4 5

Answer

-

Status : - *Marks : 0/10*

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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>

// You are using GCC
#include <iostream>
#include <vector>
using namespace std;

// Partition function for descending order
int partition(vector<char>& arr, int low, int high) {
   char pivot = arr[high];  // last element as pivot
   int i = low - 1;  // index of smaller element

   for (int j = low; j < high; ++j) {
      // Change comparison for descending order
      if (arr[j] > pivot) {
         ++i;
         swap(arr[i], arr[j]);
      }
   }

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

```
}

// Quick sort function
void quickSort(vector<char>& arr, int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        // Recursively sort elements before and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int N;
    cin >> N;

    vector<char> arr(N);
    for (int i = 0; i < N; ++i) {
        cin >> arr[i];
    }

    // Sort using quick sort (descending)
    quickSort(arr, 0, N - 1);

    // Output sorted characters
    for (char c : arr) {
        cout << c << " ";
    }
    cout << endl;

    return 0;
}
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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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>

// You are using GCC
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n;
    cin >> n;

    vector<int> arr(n);

    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }

    // Insertion Sort
    for (int i = 1; i < n; ++i) {
        int key = arr[i];
        int j = i - 1;

        // Move elements of arr[0..i-1] that are greater than key
        // to one position ahead of their current position
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }

    // Output the sorted array
    for (int i = 0; i < n; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}

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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are required to implement basic operations on a Binary Search Tree (BST), like insertion and searching.

Insertion: Given a list of integers, construct a Binary Search Tree by repeatedly inserting each integer into the tree according to the rules of a BST.

Searching: Given an integer, search for its presence in the constructed Binary Search Tree. Print whether the integer is found or not.

Write a program to calculate this efficiently.

Input Format

The first line of input consists of an integer n, representing the number of nodes in the binary search tree.

The second line consists of the values of the nodes, separated by space as integers.

The third line consists of an integer representing, the value that is to be searched.

Output Format

The output prints, "Value <value> is found in the tree." if the given value is present, otherwise it prints: "Value <value> is not found in the tree."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
8 3 10 1 6 14 23
6
Output: Value 6 is found in the tree.

Answer

```
struct Node* insertNode(struct Node* root, int value) {
   if(root==NULL){
      return createNode(value);
   }
   if(value<root->data){
      root->left=insertNode(root->left,value);
   }else if(value>root->data){
      root->right=insertNode(root->right,value);
   }
   return root;
}
struct Node* searchNode(struct Node* root, int value) {
   if(root==NULL||root->data==value){
      return root;
   }
   if(value<root->data){
      return searchNode(root->left,value);
   }
   else{
        return searchNode(root->right,value);
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is learning about Binary Search Trees (BST) in his computer science class. He wants to create a program that allows users to delete a node with a given value from a BST and print the remaining nodes using an in-order traversal.

Implement a function to help him delete a node with a given value from a BST.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the BST nodes.

The third line consists of an integer V, which is the value to delete from the BST.

Output Format

The output prints the space-separated values in the BST in an in-order traversal, after the deletion of the specified value.

If the specified value is not available in the tree, print the given input values in-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7
15

Output: 2 5 7 10

Answer

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

struct TreeNode {
   int data;
   struct TreeNode* left;
   struct TreeNode* right;
};

struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
   newNode->data = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}

struct TreeNode* insert(struct TreeNode* root, int key) {
   if (root == NULL)
      return createNode(key);
```

```
    if (key < root->data)
        root->left = insert(root->left, key);
    else if (key > root->data)
        root->right = insert(root->right, key);
    return root;
}

struct TreeNode* findMin(struct TreeNode* node) {
    while (node->left != NULL)
        node = node->left;
    return node;
}

struct TreeNode* deleteNode(struct TreeNode* root, int key) {
    if (root == NULL)
        return root;
    if (key < root->data)
        root->left = deleteNode(root->left, key);
    else if (key > root->data)
        root->right = deleteNode(root->right, key);
    else {
        if (root->left == NULL) {
            struct TreeNode* temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL) {
            struct TreeNode* temp = root->left;
            free(root);
            return temp;
        }
        struct TreeNode* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}

void inorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
int main()
{
    int N, rootValue, V;
    scanf("%d", &N);
    struct TreeNode* root = NULL;
    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }
    scanf("%d", &V);
    root = deleteNode(root, V);
    inorderTraversal(root);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In his computer science class, John is learning about Binary Search Trees (BST). He wants to build a BST and find the maximum value in the tree.

Help him by writing a program to insert nodes into a BST and find the maximum value in the tree.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the BST.

The second line consists of N space-separated integers, representing the values of the nodes to insert into the BST.

Output Format

The output prints the maximum value in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 2 7

Output: 15

Answer

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

struct TreeNode {
   int data;
   struct TreeNode* left;
   struct TreeNode* right;
};
struct TreeNode* createNode(int key) {
   struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct
TreeNode));
   newNode->data = key;
   newNode->left = newNode->right = NULL;
   return newNode;
}
// You are using GCC
struct TreeNode* insert(struct TreeNode* root, int key) {
   //Type your code here
   if(root==NULL) return createNode(key);
   if(key<root->data){
      root->left=insert(root->left,key);
   }
   else if(key>root->data){
      root->right=insert(root->right,key);
   }
   return root;
}
```

```
int findMax(struct TreeNode* root) {
    //Type your code here
    if(root==NULL) return NULL;
    if(root->right==NULL){
        return root->data;
    }
    return findMax(root->right);
}

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

    struct TreeNode* root = NULL;

    for (int i = 0; i < N; i++) {
        int key;
        scanf("%d", &key);
        if (i == 0) rootValue = key;
        root = insert(root, key);
    }

    int maxVal = findMax(root);
    if (maxVal != -1) {
        printf("%d", maxVal);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_PAH_Updated

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Yogi is working on a program to manage a binary search tree (BST) containing integer values. He wants to implement a function that removes nodes from the tree that fall outside a specified range defined by a minimum and maximum value.

Help Yogi by writing a function that achieves this.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the BST.

The second line consists of N space-separated integers, representing the elements to be inserted into the BST.

The third line consists of two space-separated integers min and max, representing the minimum value and the maximum value of the range.

Output Format

The output prints the remaining elements of the BST in an in-order traversal, after removing nodes that fall outside the specified range.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 5 15 20 12
5 15

Output: 5 10 12 15

Answer

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

struct Node {
    int key;
    struct Node* left;
    struct Node* right;
};

struct Node* newNode(int key) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->key = key;
    node->left = node->right = NULL;
    return node;
}

struct Node* insert(struct Node* root, int key) {
    if (root == NULL) return newNode(key);
    if (key < root->key)
        root->left = insert(root->left, key);
    else
        root->right = insert(root->right, key);
    return root;
}

struct Node* trimBST(struct Node* root, int min, int max) {
    if (root == NULL) return NULL;
    root->left = trimBST(root->left, min, max);
    root->right = trimBST(root->right, min, max);
    if (root->key < min) {
        struct Node* rightChild = root->right;
        free(root);
        return rightChild;
    }
    if (root->key > max) {
        struct Node* leftChild = root->left;
        free(root);
        return leftChild;
    }
    return root;
}

void inorder(struct Node* root) {
    if (root == NULL) return;
    inorder(root->left);
    printf("%d ", root->key);
    inorder(root->right);
}

int main() {
    int N;
    scanf("%d", &N);
    struct Node* root = NULL;
    for (int i = 0; i < N; i++) {
        int val;
        scanf("%d", &val);
        root = insert(root, val);
    }
    int min, max;
    scanf("%d %d", &min, &max);
    root = trimBST(root, min, max);
    inorder(root);
    printf("\n");
    return 0;
```

}

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

2. Problem Statement

Arun is exploring operations on binary search trees (BST). He wants to write a program with an unsorted distinct integer array that represents the BST keys and construct a height-balanced BST from it.

After constructing, he wants to perform the following operations that can alter the structure of the tree and traverse them using a level-order traversal:

InsertionDeletion

Your task is to assist Arun in completing the program without any errors.

Input Format

The first line of input consists of an integer N, representing the number of initial keys in the BST.

The second line consists of N space-separated integers, representing the initial keys.

The third line consists of an integer X, representing the new key to be inserted into the BST.

The fourth line consists of an integer Y, representing the key to be deleted from the BST.

Output Format

The first line of output prints "Initial BST: " followed by a space-separated list of keys in the initial BST after constructing it in level order traversal.

The second line prints "BST after inserting a new node X: " followed by a space-separated list of keys in the BST after inserting X n level order traversal.

The third line prints "BST after deleting node Y: " followed by a space-separated list of keys in the BST after deleting Y n level order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
25 14 56 28 12
34
12

Output: Initial BST: 25 14 56 12 28
BST after inserting a new node 34: 25 14 56 12 28 34
BST after deleting node 12: 25 14 56 28 34

Answer

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

typedef struct Node {
   int key;
   struct Node* left;
   struct Node* right;
} Node;

Node* createNode(int val) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->key = val;
   newNode->left = newNode->right = NULL;
   return newNode;
}

Node* insert(Node* root, int key) {
   if (root == NULL) return createNode(key);
   if (key < root->key) root->left = insert(root->left, key);
   else root->right = insert(root->right, key);
   return root;
}

Node* minValueNode(Node* node) {
   while (node && node->left) node = node->left;
```

```
    return node;
}

Node* deleteNode(Node* root, int key) {
    if (!root) return NULL;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
        root->right = deleteNode(root->right, key);
    else {
        if (!root->left) {
            Node* temp = root->right;
            free(root);
            return temp;
        }
        if (!root->right) {
            Node* temp = root->left;
            free(root);
            return temp;
        }
        Node* temp = minValueNode(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right, temp->key);
    }
    return root;
}

// Queue for level order traversal
typedef struct QueueNode {
    Node* treeNode;
    struct QueueNode* next;
} QueueNode;

typedef struct Queue {
    QueueNode *front, *rear;
} Queue;

Queue* createQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->front = q->rear = NULL;
    return q;
}
```

```
void enqueue(Queue* q, Node* node) {
    QueueNode* temp = (QueueNode*)malloc(sizeof(QueueNode));
    temp->treeNode = node;
    temp->next = NULL;
    if (!q->rear) {
        q->front = q->rear = temp;
    } else {
        q->rear->next = temp;
        q->rear = temp;
    }
}

Node* dequeue(Queue* q) {
    if (!q->front) return NULL;
    QueueNode* temp = q->front;
    Node* node = temp->treeNode;
    q->front = q->front->next;
    if (!q->front) q->rear = NULL;
    free(temp);
    return node;
}

int isEmpty(Queue* q) {
    return q->front == NULL;
}

void levelOrder(Node* root) {
    if (!root) return;
    Queue* q = createQueue();
    enqueue(q, root);
    while (!isEmpty(q)) {
        Node* curr = dequeue(q);
        printf("%d ", curr->key);
        if (curr->left) enqueue(q, curr->left);
        if (curr->right) enqueue(q, curr->right);
    }
    printf("\n");
    free(q);
}

int main() {
    int N, X, Y, i, val;
    scanf("%d", &N);
    Node* root = NULL;
    for (i = 0; i < N; ++i) {
        scanf("%d", &val);
        root = insert(root, val);
    }
    scanf("%d %d", &X, &Y);

    printf("Initial BST: ");
    levelOrder(root);

    root = insert(root, X);
    printf("BST after inserting a new node %d: ", X);
    levelOrder(root);

    root = deleteNode(root, Y);
    printf("BST after deleting node %d: ", Y);
    levelOrder(root);

    return 0;
}
```

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

3. Problem Statement

Viha, a software developer, is working on a project to automate searching for a target value in a Binary Search Tree (BST). She needs to create a program that takes an integer target value as input and determines if that value is present in the BST or not.

Write a program to assist Viha.

Input Format

The first line of input consists of integers separated by spaces, which represent the elements to be inserted into the BST. The input is terminated by entering -1.

The second line consists of an integer target, which represents the target value to be searched in the BST.

Output Format

If the target value is found in the BST, print "[target] is found in the BST".

Else, print "[target] is not found in the BST"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 3 7 1 4 6 8 -1
4
Output: 4 is found in the BST

Answer

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

typedef struct Node {
   int key;
   struct Node* left;
   struct Node* right;
} Node;

// Function to create a new node
Node* createNode(int val) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->key = val;
   newNode->left = newNode->right = NULL;
   return newNode;
}

// Insert into BST
Node* insert(Node* root, int key) {
   if (root == NULL) return createNode(key);
   if (key < root->key) root->left = insert(root->left, key);
   else root->right = insert(root->right, key);
   return root;
}
```

```
// Search in BST
int search(Node* root, int target) {
    if (!root) return 0;
    if (root->key == target) return 1;
    if (target < root->key) return search(root->left, target);
    else return search(root->right, target);
}

int main() {
    Node* root = NULL;
    int num;

    // Input until -1 is entered
    while (scanf("%d", &num) && num != -1) {
        root = insert(root, num);
    }

    int target;
    scanf("%d", &target);

    if (search(root, target)) {
        printf("%d is found in the BST\n", target);
    } else {
        printf("%d is not found in the BST\n", target);
    }

    return 0;
}
```

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

4. Problem Statement

Joseph, a computer science student, is interested in understanding binary search trees (BST) and their node arrangements. He wants to create a program to explore BSTs by inserting elements into a tree and displaying the nodes using post-order traversal of the tree.

Write a program to help Joseph implement the program.

Input Format

The first line of input consists of an integer N, representing the number of elements to insert into the BST.

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

Output Format

The output prints N space-separated integer values after the post-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 15 5 3

Output: 3 5 15 10

Answer

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

struct Node* insert(struct Node* root, int data) {
    if (root == NULL) {
        return newNode(data);
    }

    if (data < root->data) {
        root->left = insert(root->left, data);
    } else {
        root->right = insert(root->right, data);
    }

    return root;
}

void postOrder(struct Node* root) {
    if (root == NULL) {
        return;
    }

    postOrder(root->left);
    postOrder(root->right);
    printf("%d ", root->data);
}

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

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

    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        root = insert(root, elements[i]);
    }

    postOrder(root);
    printf("\n");

    return 0;
}
```

Status : Correct **_Marks : 10/10_**

5. Problem Statement

Aishu is participating in a coding challenge where she needs to reconstruct a Binary Search Tree (BST) from given preorder traversal data and then print the in-order traversal of the reconstructed BST.

Since Aishu is just learning about tree data structures, she needs your help to write a program that does this efficiently.

Input Format

The first line consists of an integer n, representing the number of nodes in the BST.

The second line of input contains n integers separated by spaces, which represent the preorder traversal of the BST.

Output Format

The output displays n space-separated integers, representing the in-order traversal of the reconstructed BST.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
10 5 1 7 40 50
Output: 1 5 7 10 40 50

Answer

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

struct Node* insert(struct Node* root, int data) {
    if (root == NULL) {
        return newNode(data);
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
    } else {
        root->right = insert(root->right, data);
    }
    return root;
}

void inorder(struct Node* root) {
    if (root == NULL) {
        return;
    }
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}

struct Node* constructBSTFromPreorder(int preorder[], int* index, int n, int min,
int max) {
    if (*index >= n) {
        return NULL;
    }

    int data = preorder[*index];
    if (data < min || data > max) {
        return NULL;
    }

    (*index)++;
    struct Node* node = newNode(data);
    node->left = constructBSTFromPreorder(preorder, index, n, min, data);
    node->right = constructBSTFromPreorder(preorder, index, n, data, max);

    return node;
}

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

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

    int index = 0;
    struct Node* root = constructBSTFromPreorder(preorder, &index, n, INT_MIN,
INT_MAX);

    inorder(root);
    printf("\n");

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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>

def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        # Merge the sorted halves
        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        # Check for any remaining elements
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1
        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

# Input handling
n = int(input())
arr = list(map(float, input().split()))

merge_sort(arr)

# Output with 3 decimal places
print(' '.join(f"{num:.3f}" for num in arr))
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: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Mike is learning about Binary Search Trees (BSTs) and wants to implement various operations on them. He wants to write a basic program for creating a BST, inserting nodes, and printing the tree in the pre-order traversal.

Write a program to help him solve this program.

Input Format

The first line of input consists of an integer N, representing the number of values to insert into the BST.

The second line consists of N space-separated integers, representing the values to insert into the BST.

Output Format

The output prints the space-separated values of the BST in the pre-order traversal.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3 1 5 2 4

Output: 3 1 2 5 4

Answer

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

struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

struct Node* createNode(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->left = newNode->right = NULL;
   return newNode;
}
// / You are using GCC
struct Node* insert(struct Node* root, int key) {
   //Type your code here
   if(root==NULL) return createNode(key);
   if(key<root->data){
      root->left=insert(root->left,key);
   }
   else if(key>root->data){
      root->right=insert(root->right,key);
   }
   return root;
}
```

```
void printPreorder(struct Node* node) {
    //Type your code here
    if(node==NULL) return;
    printf("%d ",node->data);
    printPreorder(node->left);

    printPreorder(node->right);
   // printf("%d ",node->data);
}

int main() {
    struct Node* root = NULL;

    int n;
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        root = insert(root, value);
    }

    printPreorder(root);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine a bustling coffee shop, where customers are placing their orders for their favorite coffee drinks. The cafe owner Sheeren wants to efficiently manage the queue of coffee orders using a digital system. She needs a program to handle this queue of orders.

You are tasked with creating a program that implements a queue for coffee orders. Each character in the queue represents a customer's coffee order, with 'L' indicating a latte, 'E' indicating an espresso, 'M' indicating a macchiato, 'O' indicating an iced coffee, and 'N' indicating a nabob.

Customers can place orders and enjoy their delicious coffee drinks.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Enqueue the coffee order into the queue. If the choice is 1, the following input is a space-separated character ('L', 'E', 'M', 'O', 'N').

Choice 2: Dequeue a coffee order from the queue.

Choice 3: Display the orders in the queue.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the queue:

If the choice is 1:

1. Insert the given order into the queue and display "Order for [order] is enqueued." where [order] is the coffee order that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue more orders."

If the choice is 2:

1. Dequeue a character from the queue and display "Dequeued Order: " followed by the corresponding order that is dequeued.
2. If the queue is empty without any orders, print "No orders in the queue."

If the choice is 3:

1. The output prints "Orders in the queue are: " followed by the space-separated orders present in the queue.
2. If there are no orders in the queue, print "Queue is empty. No orders available."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

Refer to the sample output for the exact text and format.

Sample Test Case

```
Input: 1 L
1 E
1 M
1 O
1 N
1 O
3
2
3
4
Output: Order for L is enqueued.
Order for E is enqueued.
Order for M is enqueued.
Order for O is enqueued.
Order for N is enqueued.
Queue is full. Cannot enqueue more orders.
Orders in the queue are: L E M O N
Dequeued Order: L
Orders in the queue are: E M O N
Exiting program
```

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

char orders[MAX_SIZE];
int front = -1;
int rear = -1;

void initializeQueue() {
   front = -1;
   rear = -1;
}

int isFull() {
    return rear == MAX_SIZE - 1;
}

int isEmpty() {
    return front == -1 || front > rear;
}

int enqueue(char order) {  //    Now returns int for use in if-condition
    if (isFull()) {
        printf("Queue is full. Cannot enqueue more orders.");
        return 0;
    } else {
        if (front == -1)
            front = 0;
        rear++;
        orders[rear] = order;
        printf("Order for %c is enqueued.", order);
        return 1;
    }
}

void dequeue() {
    if (isEmpty()) {
        printf("No orders in the queue.");
    } else {
        printf("Dequeued Order: %c", orders[front]);
        front++;
        if (front > rear) {
            front = -1;
            rear = -1;
        }
    }
}

void display() {
    if (isEmpty()) {
        printf("Queue is empty. No orders available.");
    } else {
        printf("Orders in the queue are:");
        for (int i = front; i <= rear; i++) {
            printf(" %c", orders[i]);
        }
    }
}
int main() {
    char order;
    int option;
    initializeQueue();
    while (1) {
        if (scanf("%d", &option) != 1) {
            break;
        }
        switch (option) {
            case 1:
                if (scanf(" %c", &order) != 1) {
                    break;
                }
                if (enqueue(order)) {
                }
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program");
                return 0;
            default:
                printf("Invalid option.\n");
                break;
        }
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a bustling IT department, staff regularly submit helpdesk tickets to request technical assistance. Managing these tickets efficiently is vital for providing quality support.

Your task is to develop a program that uses an array-based queue to handle and prioritize helpdesk tickets based on their unique IDs.

Implement a program that provides the following functionalities:

Enqueue Helpdesk Ticket: Add a new helpdesk ticket to the end of the queue. Provide a positive integer representing the ticket ID for the new ticket.Dequeue Helpdesk Ticket: Remove and process the next helpdesk ticket from the front of the queue. The program will display the ticket ID of the processed ticket.Display Queue: Display the ticket IDs of all the helpdesk tickets currently in the queue.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Enqueue the ticket ID into the queue. If the choice is 1, the following input is a space-separated integer, representing the ticket ID to be enqueued into the queue.

Choice 2: Dequeue a ticket from the queue.

Choice 3: Display the ticket IDs in the queue.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the queue:

If the choice is 1:

1. Insert the given ticket ID into the queue and display "Helpdesk Ticket ID [id] is enqueued." where [id] is the ticket ID that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a ticket ID from the queue and display "Dequeued Helpdesk Ticket ID: " followed by the corresponding ID that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Helpdesk Ticket IDs in the queue are: " followed by the space-separated ticket IDs present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting the program"

If any other choice is entered, print "Invalid option."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 101
1 202
1 203
1 204
1 205
1 206
3
2
3
4
Output: Helpdesk Ticket ID 101 is enqueued.
Helpdesk Ticket ID 202 is enqueued.
Helpdesk Ticket ID 203 is enqueued.
Helpdesk Ticket ID 204 is enqueued.
Helpdesk Ticket ID 205 is enqueued.
Queue is full. Cannot enqueue.
Helpdesk Ticket IDs in the queue are: 101 202 203 204 205
Dequeued Helpdesk Ticket ID: 101
Helpdesk Ticket IDs in the queue are: 202 203 204 205
Exiting the program

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

int ticketIDs[MAX_SIZE];
int front = -1;
int rear = -1;
int lastDequeued;

void initializeQueue() {
   front = -1;
   rear = -1;
}
```

```
int isFull() {
    return rear == MAX_SIZE - 1;
}

int isEmpty() {
    return front == -1 || front > rear;
}

int enqueue(int id) {
    if (isFull()) {
        printf("Queue is full. Cannot enqueue.\n");
        return 0;
    } else {
        if (front == -1)
            front = 0;
        rear++;
        ticketIDs[rear] = id;
        printf("Helpdesk Ticket ID %d is enqueued.\n", id);
        return 1;
    }
}

int dequeue() {
    if (isEmpty()) {
        return 0;
    } else {
        lastDequeued = ticketIDs[front];
        front++;
        if (front > rear) {
            front = -1;
            rear = -1;
        }
        return 1;
    }
}

void display() {
    if (isEmpty()) {
        printf("Queue is empty.\n");
    } else {
        printf("Helpdesk Ticket IDs in the queue are:");
        for (int i = front; i <= rear; i++) {
            printf(" %d", ticketIDs[i]);
        }
        printf("\n");
    }
}



int main() {
    int ticketID;
    int option;
    initializeQueue();
    while (1) {
        if (scanf("%d", &option) == EOF) {
            break;
        }
        switch (option) {
            case 1:
                if (scanf("%d", &ticketID) == EOF) {
                    break;
                }
                enqueue(ticketID);
                break;
            case 2:
                if (dequeue()) {
                    printf("Dequeued Helpdesk Ticket ID: %d\n", lastDequeued);
                } else {
                    printf("Queue is empty.\n");
                }
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting the program\n");
                return 0;
            default:
                printf("Invalid option.\n");
                break;
        }
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n, representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155
Output: 163

Answer

-

Status : Skipped *Marks : 0/10*

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 5_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John, a computer science student, is learning about binary search trees (BST) and their properties. He decides to write a program to create a BST, display it in post-order traversal, and find the minimum value present in the tree.

Help him by implementing the program.

Input Format

The first line of input consists of an integer N, representing the number of elements to insert into the BST.

The second line consists of N space-separated integers data, which is the data to be inserted into the BST.

Output Format

The first line of output prints the space-separated elements of the BST in post-order traversal.

The second line prints the minimum value found in the BST.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 3
5 10 15
Output: 15 10 5
The minimum value in the BST is: 5
```

Answer

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

struct Node {
   int data;
   struct Node* left;
   struct Node* right;
};

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}

struct Node* insert(struct Node* root, int data) {
   if(root==NULL){
      return createNode(data);
   }
   if(data<root->data){
      root->left=insert(root->left,data);
   }else if(data>root->data){
      root->right=insert(root->right,data);
```

```
    }
    return root;
}

void displayTreePostOrder(struct Node* root) {
    if(root==NULL)
        return;
    displayTreePostOrder(root->left);
    displayTreePostOrder(root->right);
    printf("%d ",root->data);
}

int findMinValue(struct Node* root) {
    if(root==NULL) return-1;
    while(root->left!=NULL){
        root=root->left;
    }
    return root->data;
}

int main() {
    struct Node* root = NULL;
    int n, data;
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        root = insert(root, data);
    }

    displayTreePostOrder(root);
    printf("\n");

    int minValue = findMinValue(root);
    printf("The minimum value in the BST is: %d", minValue);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mathivanan P
Email: 240801197@rajalakshmi.edu.in
Roll no: 240801197
Phone: 9080427752
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters.Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

Input Format

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

Output Format

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -

Output: Forward Playlist: a b c
Backward Playlist: c b a

Answer

-

Status : Skipped ***Marks : 0/10***