Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING

AND INFORMATION SYSTEMS

FALL SEMESTER 2024-2025

PMCA501P – DATA STRUCTURES AND ALGORITHM LAB

DIGITAL ASSIGNMENT – 2

SUBMITTED ON: 10 – OCT - 2024

SUBMITTED BY-

AKASH KUMAR BANIK

PROGRAM: MCA

REGISTER No.: 24MCA0242

PROBLEM STATEMENT – 1:

1. Look into the following student structure definition.

```
Struct Student

{

Char stud_name[60];

Char reg_num[10];

int mark1;

int mark2;

int mark3;

int total;

float average;

char grade;

};
```

Assume marks are out of 100. The absolute grading shall be obtained based on the following assumptions

91 to 100 - S grade

81 to 90 - A grade

71 to 81 - B grade

61 to 70 - C grade

51 to 60 - D grade

41 – 50 - E grade

Less than 40 - F grade- Fail

Not appeared or Not eligible or Debarred - N grade

Write a C program to perform the following operations as per the Singly Linked List principle.

a) Create a new node and add further nodes at the end of the list. Also ensure the duplication should not occur while adding a node as per the primary field register number.

b) Traverse the nodes from first node to last node and print the current list of nodes.

c) Print from the last node to first node along with a help of another suitable data structure.

d) Find the nodes whose student average are greater than 90 and delete those nodes from the current list and move those nodes to a new list and print the new list.

e) Also traverse and print the remaining original list after performing step d.

f) Count the number of times the head node value is updated among the performed insertions at various positions. Print that number.

g) Count the number of times the head node value is updated among the performed deletions at various positions. Print that number.

CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Student {

    char stud_name[50];

    char reg_num[10];

    int mark1, mark2, mark3, total;

    float average;

    char grade;

    struct Student *next;

};

typedef struct Student Student;

Student *head = NULL;

Student *new_list = NULL;
```

```
int head_insertions = 0;

int head_deletions = 0;


void calculate(Student *node) {

    int m1 = (node->mark1 == -1) ? 0 : node->mark1;

    int m2 = (node->mark2 == -1) ? 0 : node->mark2;

    int m3 = (node->mark3 == -1) ? 0 : node->mark3;

    node->total = m1 + m2 + m3;

    node->average = node->total / 3.0;


    if (node->mark1 == -1 || node->mark2 == -1 || node->mark3 == -1) {

        node->grade = 'N';

    } else if (node->average >= 91) {

        node->grade = 'S';

    } else if (node->average >= 81) {

        node->grade = 'A';

    } else if (node->average >= 71) {

        node->grade = 'B';

    } else if (node->average >= 61) {

        node->grade = 'C';

    } else if (node->average >= 51) {

        node->grade = 'D';

    } else if (node->average >= 41) {

        node->grade = 'E';

    } else {
        node->grade = 'F';
    }
}

int is_duplicate(char reg_num[]) {
    Student *temp = head;
    while (temp) {
        if (strcmp(temp->reg_num, reg_num) == 0) {
            return 1;
        }
        temp = temp->next;
    }
    return 0;
}
```

// a) Create new node and add it to the end of the list, avoiding duplicates

```
void add_node(char name[], char reg[], int m1, int m2, int m3) {
    if (is_duplicate(reg)) {
        printf("\nDuplicate registration number. Node not added.\n");
        return;
    }
    Student *new_node = (Student *)malloc(sizeof(Student));
    strcpy(new_node->stud_name, name);
    strcpy(new_node->reg_num, reg);
    new_node->mark1 = m1;
    new_node->mark2 = m2;
    new_node->mark3 = m3;
    calculate(new_node);
    new_node->next = NULL;
    if (!head) {
        head = new_node;
        head_insertions++;
    } else {
        Student *temp = head;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = new_node;
    }
}
```

// **b) Traverse and print the list**

```
void traverse_list(Student *node) {
    Student *temp = node;
    if (!temp) {
        printf("List is empty.\n");
        return;
    }
    while (temp) {
        printf("Name: %s, Reg No: %s, Total: %d, Average: %.2f, Grade: %c\n",
            temp->stud_name, temp->reg_num, temp->total, temp->average, temp->grade);
        temp = temp->next;
    }
}
```

// c) Print from last to first using another data structure (stack)

```
void print_reverse() {
    if (!head) {
        printf("\nList is empty.\n");
        return;
    }
    Student *temp = head;
    int count = 0;
    Student *stack[100];
    while (temp) {
        stack[count++] = temp;
        temp = temp->next;
    }
    for (int i = count - 1; i >= 0; i--) {
        printf("Name: %s, Reg No: %s, Total: %d, Average: %.2f, Grade: %c\n",
            stack[i]->stud_name, stack[i]->reg_num, stack[i]->total, stack[i]->average, stack[i]-
>grade);
    }
}
```

// d) Move students with average > 90 to a new list and delete from original list

```
void move_high_achievers() {
    Student *prev = NULL, *temp = head;
    while (temp) {
        if (temp->average > 90) {
            Student *to_move = temp;
            if (prev) {
                prev->next = temp->next;
            } else {
                head = temp->next;
                head_deletions++;
            }
            temp = temp->next;
            to_move->next = new_list;
            new_list = to_move;
        } else {
            prev = temp;
            temp = temp->next;
        }
    }
}
```

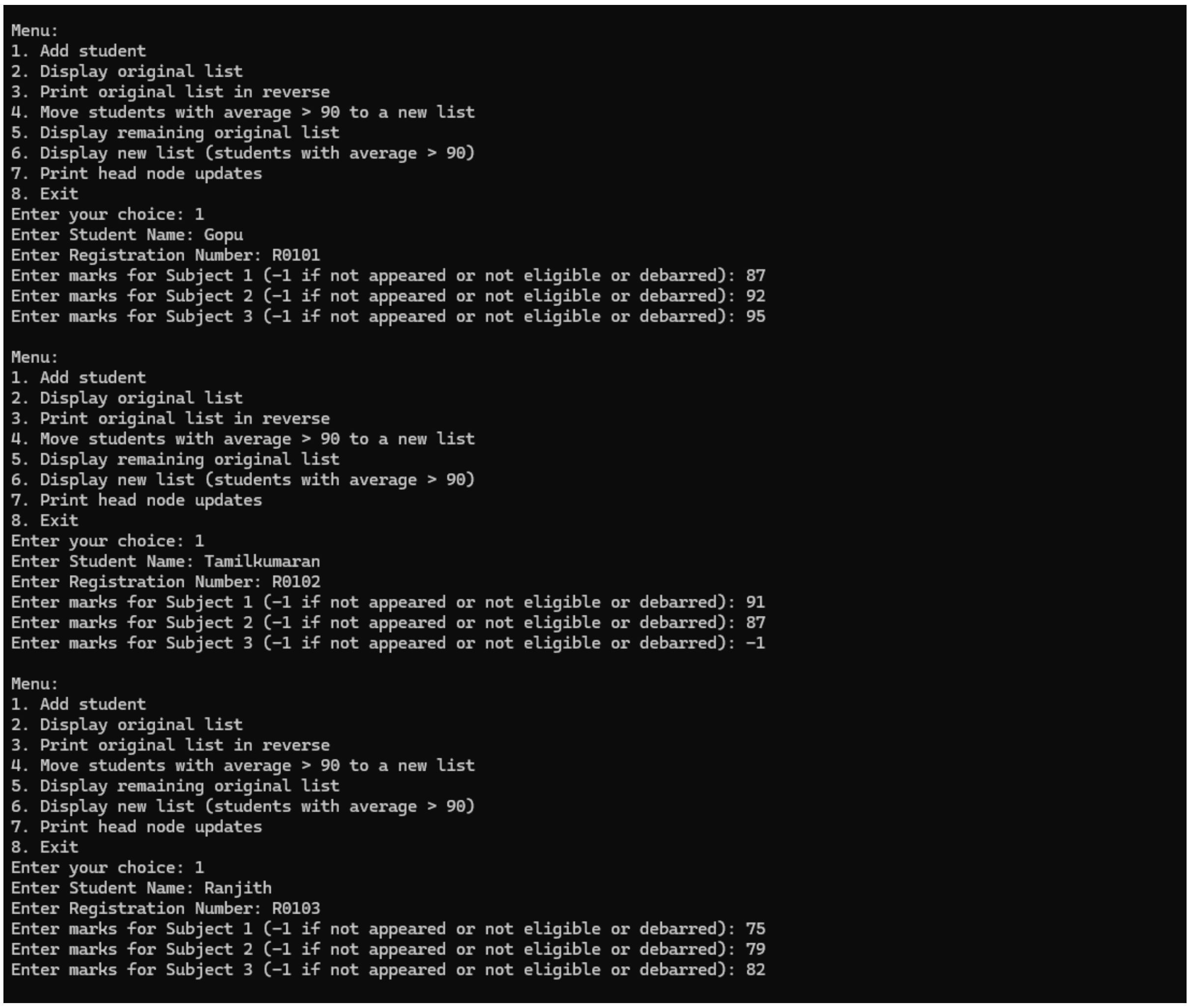
// f) & g) Print the number of head insertions and deletions

```
void print_head_updates() {
    printf("Head insertions: %d\n", head_insertions);
    printf("Head deletions: %d\n", head_deletions);
}


void get_student_input() {
    char name[60], reg[10];
    int m1, m2, m3;
    printf("Enter Student Name: ");
    scanf("%s", name);
    printf("Enter Registration Number: ");
    scanf("%s", reg);
    printf("Enter marks for Subject 1 (-1 if not appeared or not eligible or debarred): ");
    scanf("%d", &m1);
    printf("Enter marks for Subject 2 (-1 if not appeared or not eligible or debarred): ");
    scanf("%d", &m2);
    printf("Enter marks for Subject 3 (-1 if not appeared or not eligible or debarred): ");
    scanf("%d", &m3);
    add_node(name, reg, m1, m2, m3);
}


int main() {
    int choice;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Add student\n");
        printf("2. Display original list\n");
        printf("3. Print original list in reverse\n");
        printf("4. Move students with average > 90 to a new list\n");
        printf("5. Display remaining original list\n");
        printf("6. Display new list (students with average > 90)\n");
        printf("7. Print head node updates\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
          case 1:
            get_student_input();
            break;
          case 2:
            printf("Original list:\n");
            traverse_list(head);
            break;
          case 3:
            printf("Original list in reverse:\n");
            print_reverse();
            break;
          case 4:
            move_high_achievers();
            printf("Moved below students with average > 90 to new list.\n");
            traverse_list(new_list);
            break;
```

```
        case 5:

          // e) Also traverse and print the remaining original list after performing step d.

          printf("Remaining original list students with average < 90:\n");

          traverse_list(head);

          break;

        case 6:

          printf("New list (students with average > 90):\n");

          traverse_list(new_list);

          break;

        case 7:

          print_head_updates();

          break;

        case 8:

          exit(0);

        default:

          printf("\nInvalid choice. Please try again.\n");

      }

    }

  return 0;

}
```

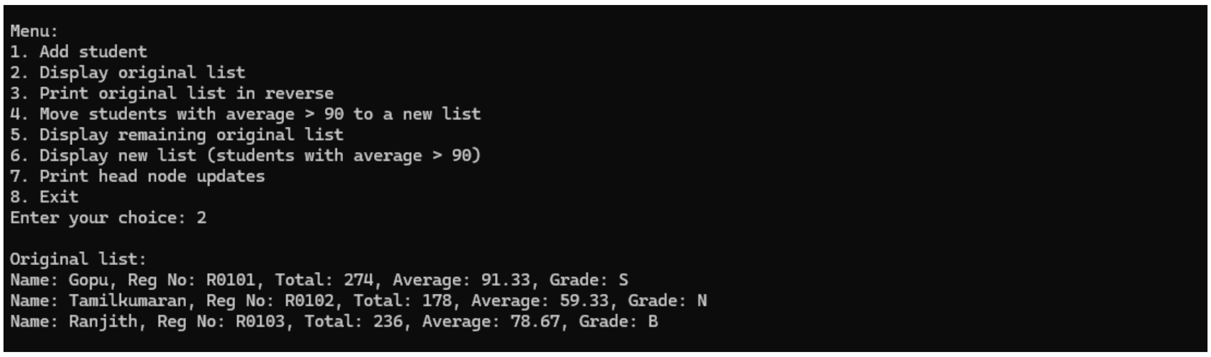
OUTPUT:

a)



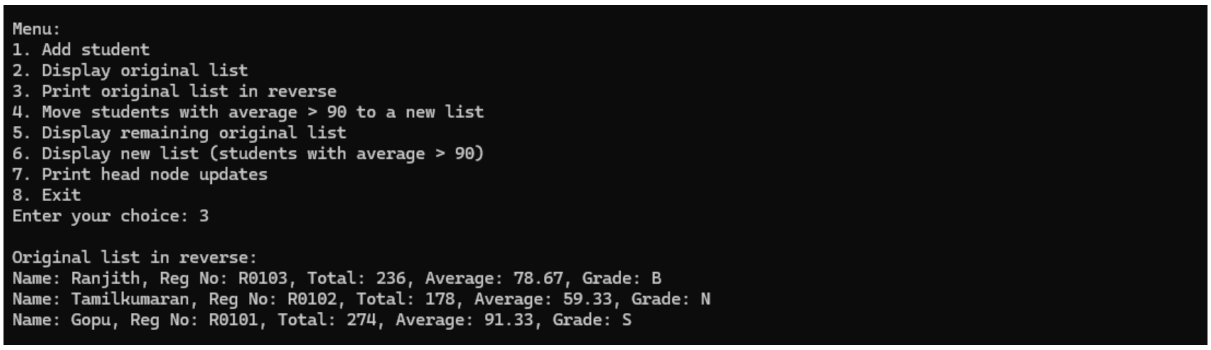
```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 1
Enter Student Name: Gopu
Enter Registration Number: R0101
Enter marks for Subject 1 (-1 if not appeared or not eligible or debarred): 87
Enter marks for Subject 2 (-1 if not appeared or not eligible or debarred): 92
Enter marks for Subject 3 (-1 if not appeared or not eligible or debarred): 95

Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 1
Enter Student Name: Tamilkumaran
Enter Registration Number: R0102
Enter marks for Subject 1 (-1 if not appeared or not eligible or debarred): 91
Enter marks for Subject 2 (-1 if not appeared or not eligible or debarred): 87
Enter marks for Subject 3 (-1 if not appeared or not eligible or debarred): -1

Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 1
Enter Student Name: Ranjith
Enter Registration Number: R0103
Enter marks for Subject 1 (-1 if not appeared or not eligible or debarred): 75
Enter marks for Subject 2 (-1 if not appeared or not eligible or debarred): 79
Enter marks for Subject 3 (-1 if not appeared or not eligible or debarred): 82
```

b)



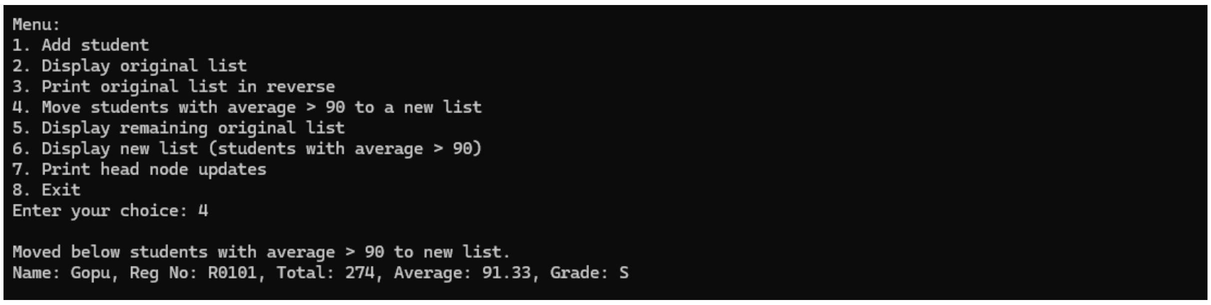
```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 2

Original list:
Name: Gopu, Reg No: R0101, Total: 274, Average: 91.33, Grade: S
Name: Tamilkumaran, Reg No: R0102, Total: 178, Average: 59.33, Grade: N
Name: Ranjith, Reg No: R0103, Total: 236, Average: 78.67, Grade: B
```

c)



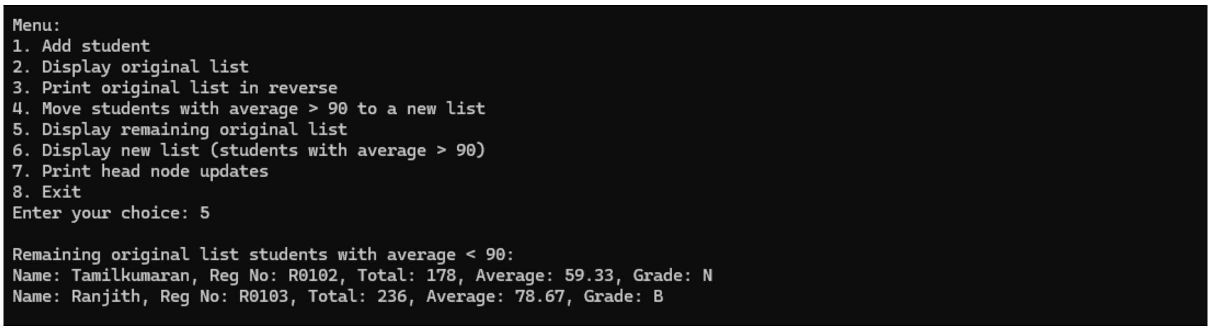
```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 3

Original list in reverse:
Name: Ranjith, Reg No: R0103, Total: 236, Average: 78.67, Grade: B
Name: Tamilkumaran, Reg No: R0102, Total: 178, Average: 59.33, Grade: N
Name: Gopu, Reg No: R0101, Total: 274, Average: 91.33, Grade: S
```

d)



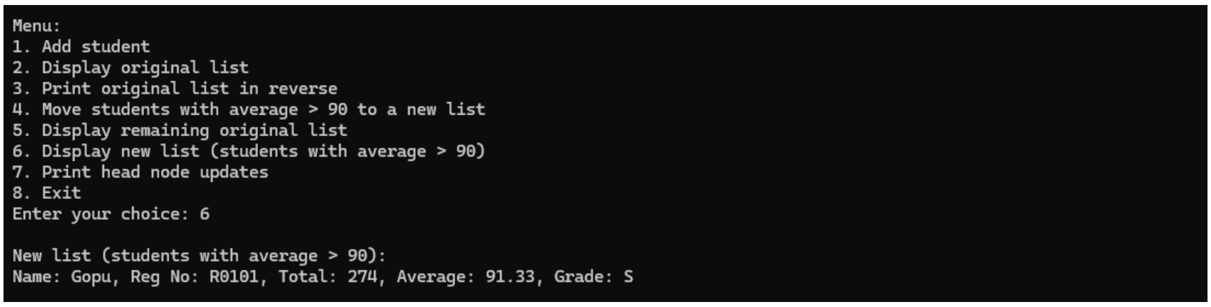
```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 4

Moved below students with average > 90 to new list.
Name: Gopu, Reg No: R0101, Total: 274, Average: 91.33, Grade: S
```

e)



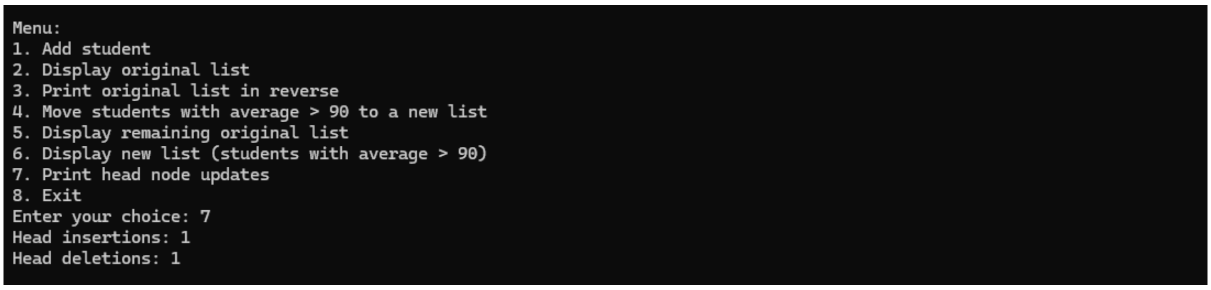
```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 5

Remaining original list students with average < 90:
Name: Tamilkumaran, Reg No: R0102, Total: 178, Average: 59.33, Grade: N
Name: Ranjith, Reg No: R0103, Total: 236, Average: 78.67, Grade: B
```

f)



```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 6

New list (students with average > 90):
Name: Gopu, Reg No: R0101, Total: 274, Average: 91.33, Grade: S
```

g)



```
Menu:
1. Add student
2. Display original list
3. Print original list in reverse
4. Move students with average > 90 to a new list
5. Display remaining original list
6. Display new list (students with average > 90)
7. Print head node updates
8. Exit
Enter your choice: 7
Head insertions: 1
Head deletions: 1
```

PROBLEM STATEMENT – 2:

2. Let us assume the following sequence of input

7,7,5,2,5,2,8,5,6,8,8,7,4,4,7,2,3,8,9,3,2,9,7,8,3

Count the number of occurrence of each digit (as per increasing order of given digit) and store those numbers in a new array. The result (intermediate output sequence) of this process looks like 4,3,2,3,1,5,5,2

Means digit 2 occurs 4 times, digit 3 occurs 3 times, digit 4 occurs 2 times, digit 5 occurs 3 times, digit 6 occurs 1 time, digit 7 occurs 5 times, digit 8 occurs 5 times, and digit 9 occurs 2 times.

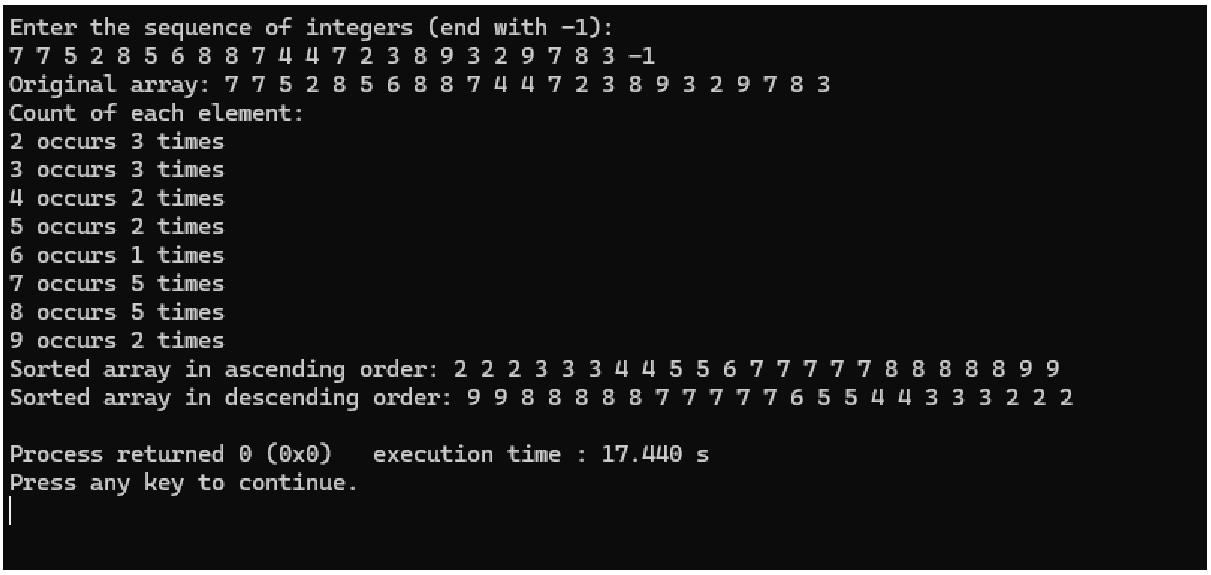
Assume this intermediate output sequence as an input for merge sort and perform merge sort to get sorted order in both ascending and descending order.

CODE:

```
#include <stdio.h>

#include <stdlib.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[n1], R[n2];

    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);
    }
}

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

void reverseArray(int arr[], int size) {
    int start = 0;
    int end = size - 1;
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

void countOccurrences(int arr[], int size) {
    int count[100] = {0};
    for (int i = 0; i < size; i++) {
        count[arr[i]]++;
    }

    printf("Count of each element:\n");
    for (int i = 0; i < 100; i++) {
        if (count[i] > 0) {
            printf("%d occurs %d times\n", i, count[i]);
        }
    }
}
int main() {
    int arr[100], size = 0, temp;
    printf("Enter the sequence of integers (end with -1):\n");
    while (scanf("%d", &temp) && temp != -1) {
        arr[size++] = temp;
    }
    printf("Original array: ");
    printArray(arr, size);
    countOccurrences(arr, size);
    mergeSort(arr, 0, size - 1);
    printf("Sorted array in ascending order: ");
    printArray(arr, size);
    reverseArray(arr, size);
    printf("Sorted array in descending order: ");
    printArray(arr, size);
    return 0;
}
```

OUTPUT:



```
Enter the sequence of integers (end with -1):
7 7 5 2 8 5 6 8 8 7 4 4 7 2 3 8 9 3 2 9 7 8 3 -1
Original array: 7 7 5 2 8 5 6 8 8 7 4 4 7 2 3 8 9 3 2 9 7 8 3
Count of each element:
2 occurs 3 times
3 occurs 3 times
4 occurs 2 times
5 occurs 2 times
6 occurs 1 times
7 occurs 5 times
8 occurs 5 times
9 occurs 2 times
Sorted array in ascending order: 2 2 2 3 3 3 4 4 5 5 6 7 7 7 7 7 8 8 8 8 8 9 9
Sorted array in descending order: 9 9 8 8 8 8 8 7 7 7 7 7 6 5 5 4 4 3 3 3 2 2 2

Process returned 0 (0x0)   execution time : 17.440 s
Press any key to continue.
```

PROBLEM STATEMENT – 3:

Assume the following input sequence (Must be an unrepeated sequence, ask the user to re-enter if the sequence is with repeated values – and check this as a validation while giving input) be 4,3,2,1,5,8. Using this input sequence find the set of numbers which is divisible by any even number and repeat the till the end of the given array and move all the obtained intermediate results into a new array. In this case we got the following intermediate results for the given unrepeated input sequence 4,2,8. (order of this intermediate result is also important, means the given input sequence must be tracked from left to right). Using this intermediate result to search for the second smallest index cum number using both linear search and binary search (also try with a lengthier sequence of valid input).

This gives the final answer as: index is 1 and the corresponding number is 2.

Note: Assume index ranges from 0 to n-1.

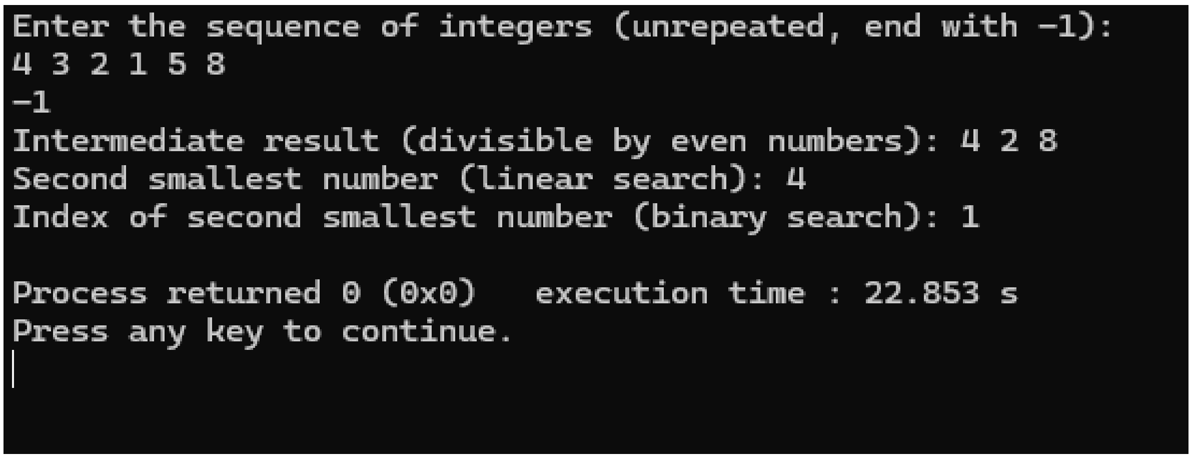
CODE:

```
#include <stdio.h>

#include <stdlib.h>


void validateInput(int arr[], int size) {

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

        for (int j = i + 1; j < size; j++) {

            if (arr[i] == arr[j]) {

                printf("Input contains repeated values. Please enter a valid sequence.\n");

                exit(0);

            }

        }

    }

}
```

```
int isDivisibleByEven(int num) {
    return num % 2 == 0;
}


void findDivisibleByEven(int arr[], int size, int result[], int *resSize) {
    for (int i = 0; i < size; i++) {
        if (isDivisibleByEven(arr[i])) {
            result[(*resSize)++] = arr[i];
        }
    }
}


int findSecondSmallestLinear(int arr[], int size) {
    int firstSmallest, secondSmallest;
    firstSmallest = secondSmallest = __INT_MAX__;
    for (int i = 0; i < size; i++) {
        if (arr[i] < firstSmallest) {
            secondSmallest = firstSmallest;
            firstSmallest = arr[i];
        } else if (arr[i] < secondSmallest && arr[i] != firstSmallest) {
            secondSmallest = arr[i];
        }
    }
    return secondSmallest;
}
```

```
int binarySearchSecondSmallest(int arr[], int low, int high, int secondSmallest) {
    int mid, index = -1;
    while (low <= high) {
        mid = low + (high - low) / 2;
        if (arr[mid] == secondSmallest) {
            index = mid;
            break;
        } else if (arr[mid] < secondSmallest) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return index;
}

void sortArray(int arr[], int size) {
    int temp;
    for (int i = 0; i < size - 1; i++) {
        for (int j = i + 1; j < size; j++) {
            if (arr[i] > arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

int main() {
    int arr[100], size = 0, temp;
    int result[100], resSize = 0;
    printf("Enter the sequence of integers (unrepeated, end with -1):\n");
    while (scanf("%d", &temp) && temp != -1) {
        arr[size++] = temp;
    }
    validateInput(arr, size);
    findDivisibleByEven(arr, size, result, &resSize);
    if (resSize < 2) {
        printf("Not enough even divisible numbers to find second smallest.\n");
        return 0;
    }
    printf("Intermediate result (divisible by even numbers): ");
    for (int i = 0; i < resSize; i++) {
        printf("%d ", result[i]);
    }
    printf("\n");
    int secondSmallest = findSecondSmallestLinear(result, resSize);
    printf("Second smallest number (linear search): %d\n", secondSmallest);
    sortArray(result, resSize);

    int index = binarySearchSecondSmallest(result, 0, resSize - 1, secondSmallest);

    printf("Index of second smallest number (binary search): %d\n", index);

    return 0;

}
```

OUTPUT:



```
Enter the sequence of integers (unrepeated, end with -1):
4 3 2 1 5 8
-1
Intermediate result (divisible by even numbers): 4 2 8
Second smallest number (linear search): 4
Index of second smallest number (binary search): 1

Process returned 0 (0x0)   execution time : 22.853 s
Press any key to continue.
```