**DSA-LAB ASSIGNMENT 3**

**(Name:Sneha Kondawar)**

**(Roll No:BT22CSE075)**

**Question 1 and 2:**

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <limits.h>

#define max(a, b) ((a) > (b) ? (a) : (b))

struct Node

{

    int val;

    struct Node \*next;

};

int findMax(int a[], int n)

{

    int m = INT\_MIN;

    for (int i = 0; i < n; i++)

    {

        m = max(a[i], m);

    }

    return m;

}

int countDigits(int x)

{

    int c = 0;

    while (x > 0)

    {

        x /= 10;

        c++;

    }

    return c;

}

void Insert(struct Node \*\*b, int value, int idx)

{

    struct Node \*temp = (struct Node \*)malloc(sizeof(struct Node));

    temp->val = value;

    temp->next = NULL;

    if (b[idx] == NULL)

    {

        b[idx] = temp;

    }

    else

    {

        struct Node \*p = b[idx];

        while (p->next != NULL)

        {

            p = p->next;

        }

        p->next = temp;

    }

}

int Delete(struct Node \*\*b, int idx)

{

    struct Node \*p = b[idx];

    b[idx] = b[idx]->next;

    int x = p->val;

    free(p);

    return x;

}

void radixSort(int a[], int n)

{

    int m = findMax(a, n);

    int nPass = countDigits(m);

    struct Node \*\*pos = (struct Node \*\*)malloc(sizeof(struct Node) \* 10);

    struct Node \*\*neg = (struct Node \*\*)malloc(sizeof(struct Node) \* 10);

    for (int i = 0; i < 10; i++) pos[i] = NULL;

    for (int i = 0; i < 10; i++) neg[i] = NULL;

    for (int i = 0; i < nPass; i++)

    {

        for (int j = 0; j < n; j++)

        {

            int binIdx;

            if (a[j] >= 0) {

                binIdx = (int)(a[j] / pow(10, i)) % 10;

                Insert(pos, a[j], binIdx);

            } else {

                binIdx = 9 - abs((int)(a[j] / pow(10, i)) % 10);

                Insert(neg, a[j], binIdx);

            }

        }

        int j = 0;

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

            while (neg[i] != NULL) {

                a[j++] = Delete(neg, i);

            }

        }

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

            while (pos[i] != NULL) {

                a[j++] = Delete(pos, i);

            }

        }

    }

    free(pos);

    free(neg);

}

int main()

{

    int a[] = {11, 13, -7, 12, 16, 9, 24, 5, 10, 3, -25, -18, -3}, n = 13;

    radixSort(a, n);

    for (int i = 0; i < n; i++)

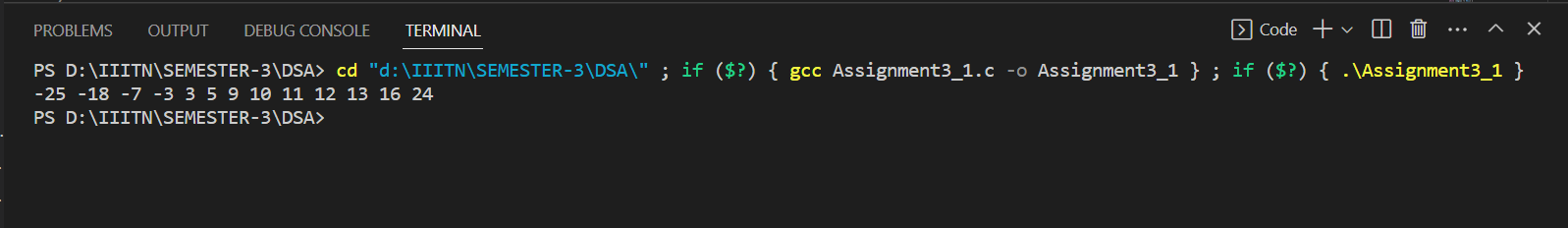
        printf("%d ", a[i]);

    printf("\n");

    return 0;

}

OUTPUT:



QUESTION 3:

#include <stdio.h>

#include <stdlib.h>

struct Node {

    int row, col, val;

    struct Node\* next;

};

void Display(struct Node\* head) {

    struct Node\* p = head;

    while (p != NULL) {

        printf("Row: %d, Column: %d, Value: %d\n", p->row + 1, p->col + 1, p->val);

        p = p->next;

    }

}

void Insert(struct Node\*\* head, int row, int col, int val) {

    struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));

    temp->row = row;

    temp->col = col;

    temp->val = val;

    if (\*head == NULL || row < (\*head)->row || (row == (\*head)->row && col < (\*head)->col)) {

        temp->next = \*head;

        \*head = temp;

    } else {

        struct Node\* c = \*head;

        while (c->next != NULL && (row > c->next->row || (row == c->next->row && col > c->next->col))) {

            c = c->next;

        }

        temp->next = c->next;

        c->next = temp;

    }

}

int main() {

    int r = 3, c = 4;

    int m[r][c];

    m[0][0] = 0; m[0][1] = 0; m[0][2] = 0; m[0][3] = 0;

    m[1][0] = 1; m[1][1] = 5; m[1][2] = 0; m[1][3] = 0;

    m[2][0] = 0; m[2][1] = 0; m[2][2] = 0; m[2][3] = 8;

    struct Node\* head = NULL;

    for (int i = 0; i < r; i++) {

        for (int j = 0; j < c; j++) {

            if (m[i][j] != 0) {

                Insert(&head, i, j, m[i][j]);

            }

        }

    }

    printf("Linked List Representation of Sparse Matrix:\n");

    Display(head);

    while (head != NULL) {

        struct Node\* temp = head;

        head = head->next;

        free(temp);

    }

    return 0;

}

OUTPUT:

