Assignment 9

DSA LAB

2029196

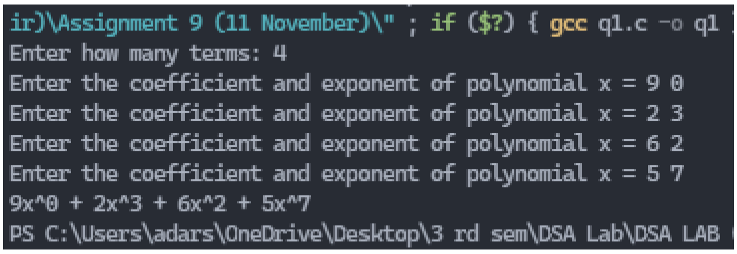
Adarsh Kumar

Q1 WAP to create a linked list that represents a polynomial expression with single variable (i.e. 5x7-3x5+x2+9) and display the polynomial by using user defined functions for creation and display.

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

struct node
{
    int coeff;
    int exp;
    struct node *next;
};

void create(struct node **head)
{
    int noOfTerms;
    printf("Enter how many terms: ");
    scanf("%d", &noOfTerms);

    struct node *ptr, *current;

    for (int i = 0; i < noOfTerms; i++)
    {
        current = malloc(sizeof(struct node));
        current->next = NULL;
        printf("Enter the coefficient and exponent of polynomial x =
");
        scanf("%d%d", &current->coeff, &current->exp);
        if (*head == NULL)
        {
            *head = current;
            ptr = current;
        }
        else
        {
            ptr->next = current;
```

```c
            ptr = current;
        }
    }
}

void display(struct node *head)
{
    struct node *current = head;
    while (current != NULL)
    {
        printf("%dx^%d", current->coeff, current->exp);
        current = current->next;
        if (current->coeff >= 0)
        {
            if (current != NULL)
                printf(" + ");
        }
        else
            printf(" ");
    }
}

int main()
{
    struct node *h1;
    h1 = NULL;

    create(&h1);
    display(h1);

    return 0;
}
```

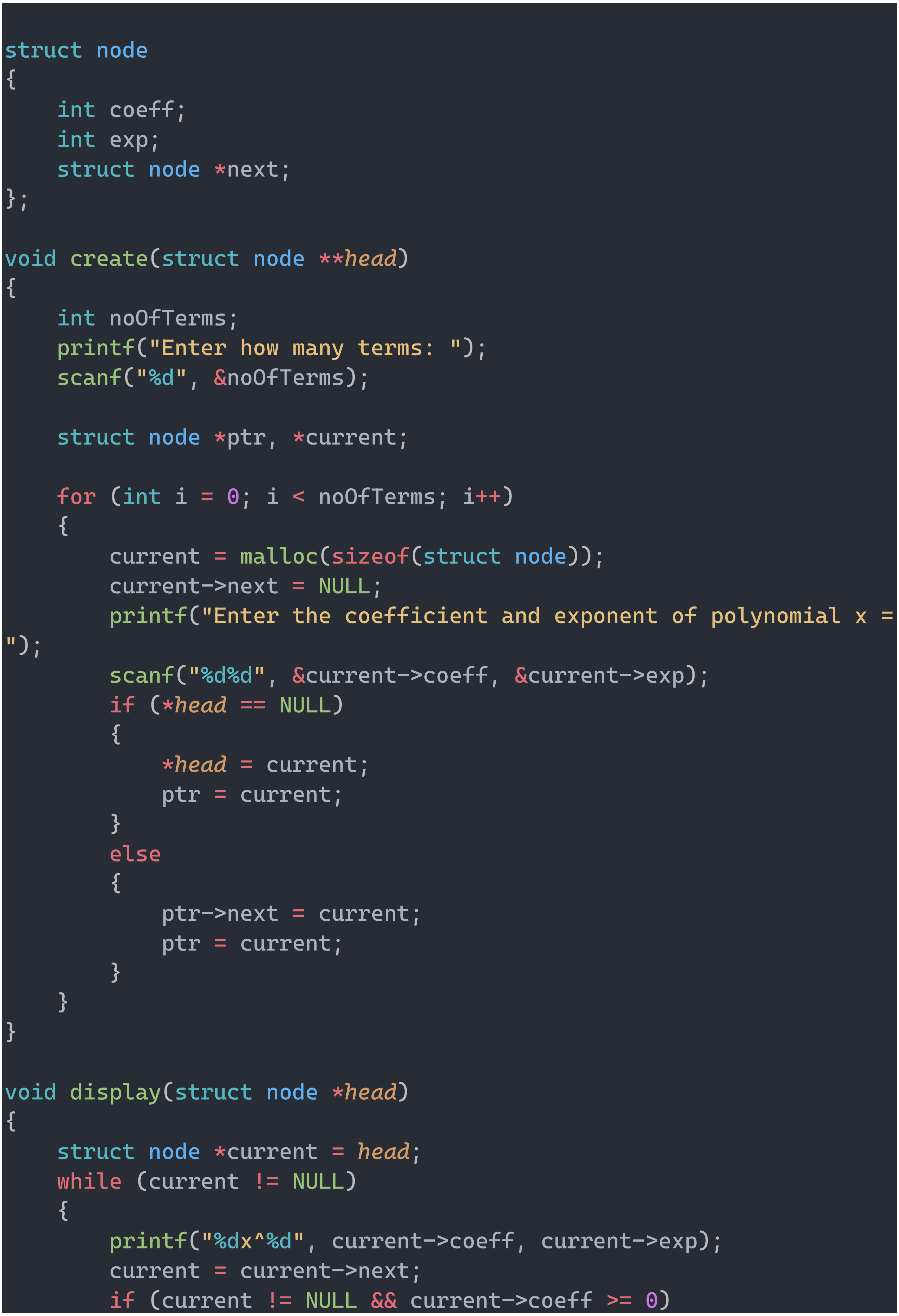
OUTPUT:-



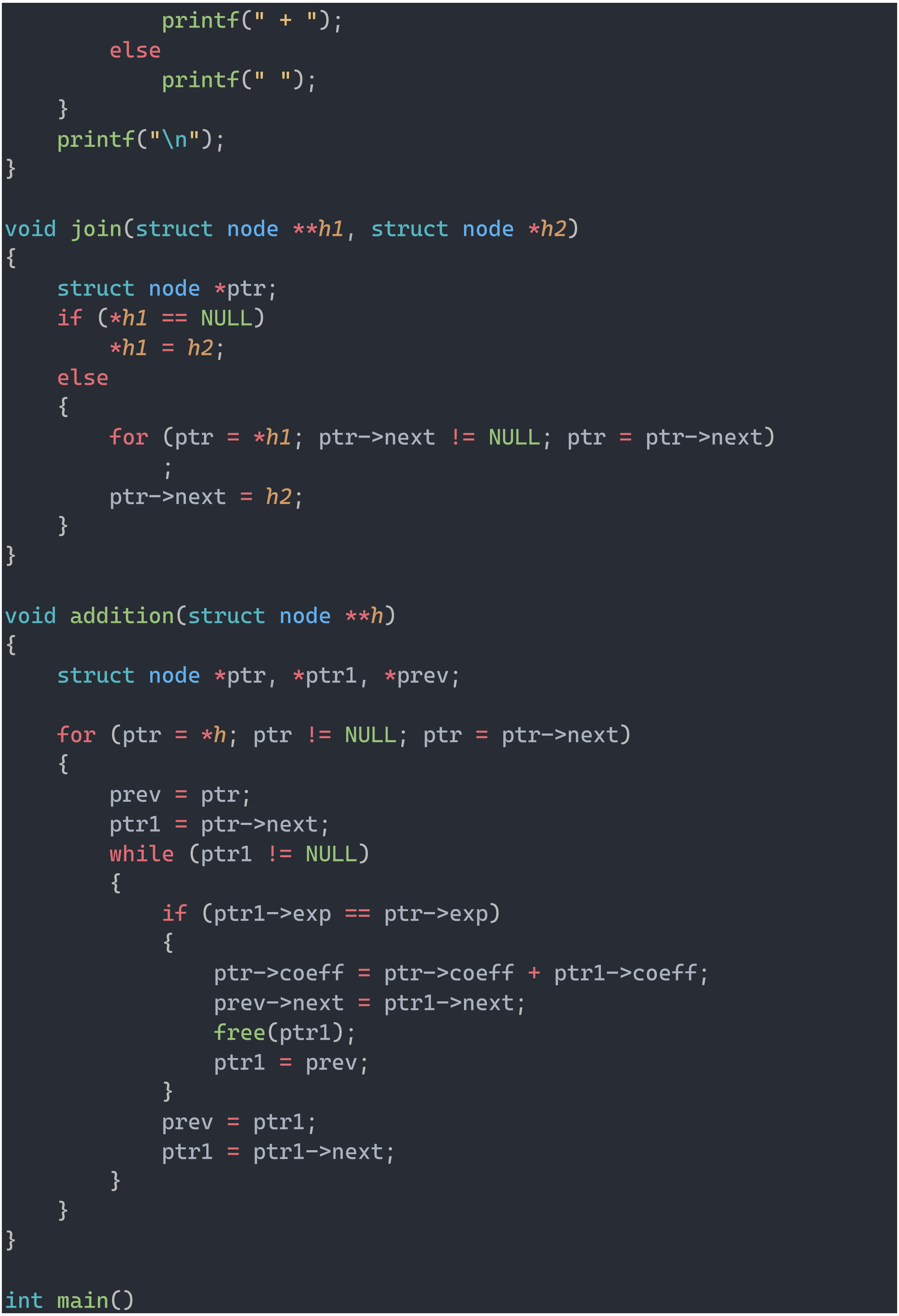
```
ir)\Assignment 9 (11 November)\" ; if ($?) { gcc q1.c -o q1
Enter how many terms: 4
Enter the coefficient and exponent of polynomial x = 9 0
Enter the coefficient and exponent of polynomial x = 2 3
Enter the coefficient and exponent of polynomial x = 6 2
Enter the coefficient and exponent of polynomial x = 5 7
9x^0 + 2x^3 + 6x^2 + 5x^7
PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DSA LAB
```

Q2. WAP by modifying the first program to add two polynomials with single variable. Use the same function in first prog. written for creation & display operations and write a new function for addition operations.

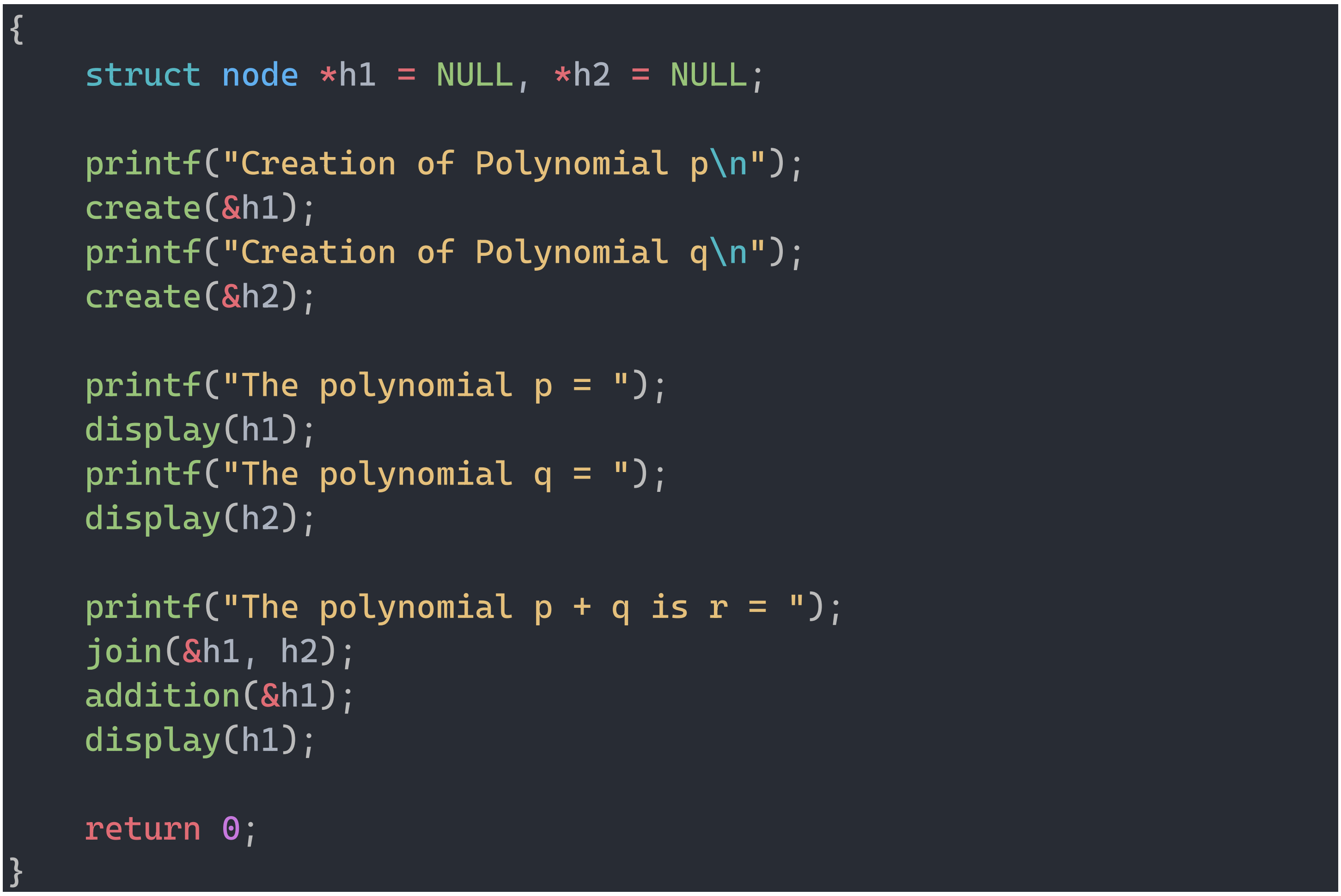
```c
#include <stdio.h>
#include <stdlib.h>
```



```c
struct node
{
    int coeff;
    int exp;
    struct node *next;
};

void create(struct node **head)
{
    int noOfTerms;
    printf("Enter how many terms: ");
    scanf("%d", &noOfTerms);

    struct node *ptr, *current;

    for (int i = 0; i < noOfTerms; i++)
    {
        current = malloc(sizeof(struct node));
        current->next = NULL;
        printf("Enter the coefficient and exponent of polynomial x = ");
        scanf("%d%d", &current->coeff, &current->exp);
        if (*head == NULL)
        {
            *head = current;
            ptr = current;
        }
        else
        {
            ptr->next = current;
            ptr = current;
        }
    }
}

void display(struct node *head)
{
    struct node *current = head;
    while (current != NULL)
    {
        printf("%dx^%d", current->coeff, current->exp);
        current = current->next;
        if (current != NULL && current->coeff >= 0)
```

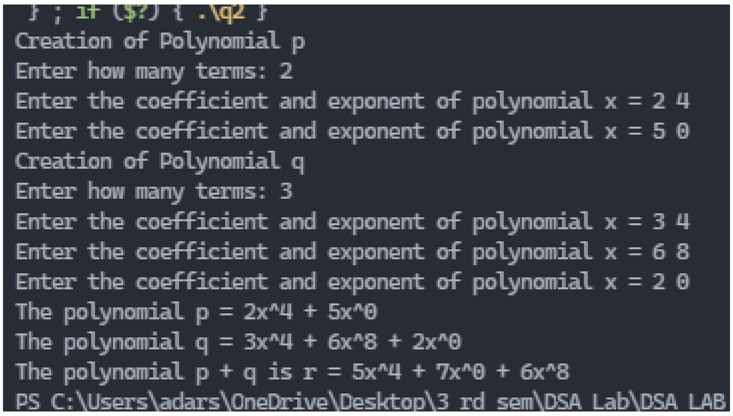


```
            printf(" + ");
        else
            printf(" ");
    }
    printf("\n");
}

void join(struct node **h1, struct node *h2)
{
    struct node *ptr;
    if (*h1 == NULL)
        *h1 = h2;
    else
    {
        for (ptr = *h1; ptr->next != NULL; ptr = ptr->next)
            ;
        ptr->next = h2;
    }
}

void addition(struct node **h)
{
    struct node *ptr, *ptr1, *prev;

    for (ptr = *h; ptr != NULL; ptr = ptr->next)
    {
        prev = ptr;
        ptr1 = ptr->next;
        while (ptr1 != NULL)
        {
            if (ptr1->exp == ptr->exp)
            {
                ptr->coeff = ptr->coeff + ptr1->coeff;
                prev->next = ptr1->next;
                free(ptr1);
                ptr1 = prev;
            }
            prev = ptr1;
            ptr1 = ptr1->next;
        }
    }
}

int main()
```



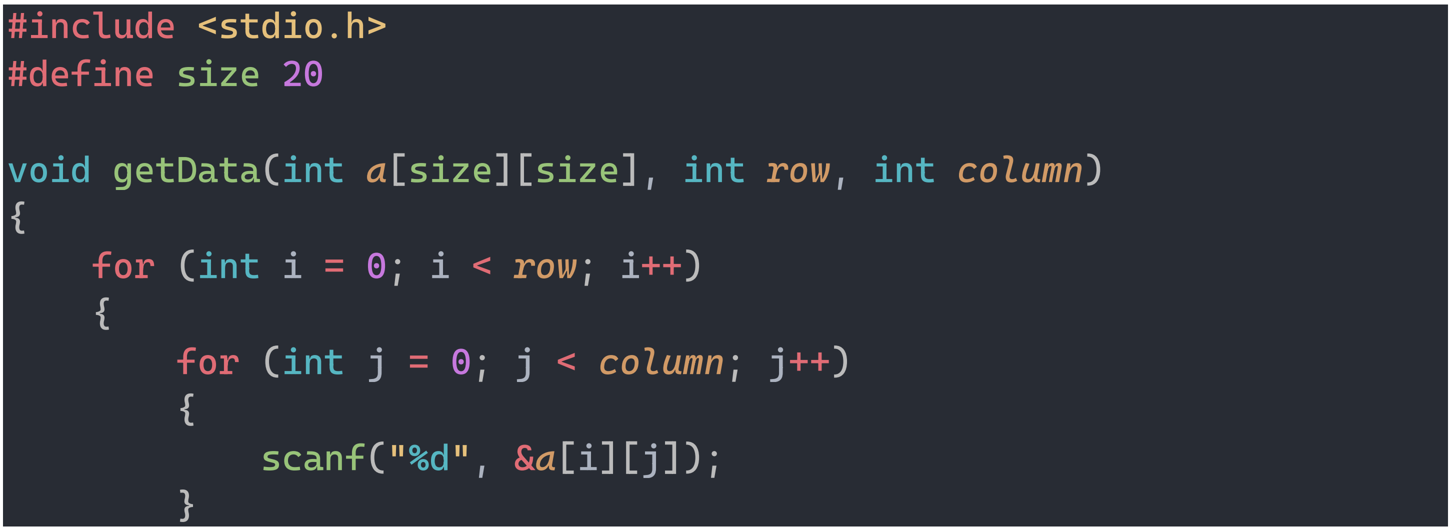
```
{
    struct node *h1 = NULL, *h2 = NULL;

    printf("Creation of Polynomial p\n");
    create(&h1);
    printf("Creation of Polynomial q\n");
    create(&h2);

    printf("The polynomial p = ");
    display(h1);
    printf("The polynomial q = ");
    display(h2);

    printf("The polynomial p + q is r = ");
    join(&h1, h2);
    addition(&h1);
    display(h1);

    return 0;
}
```

OUTPUT:-

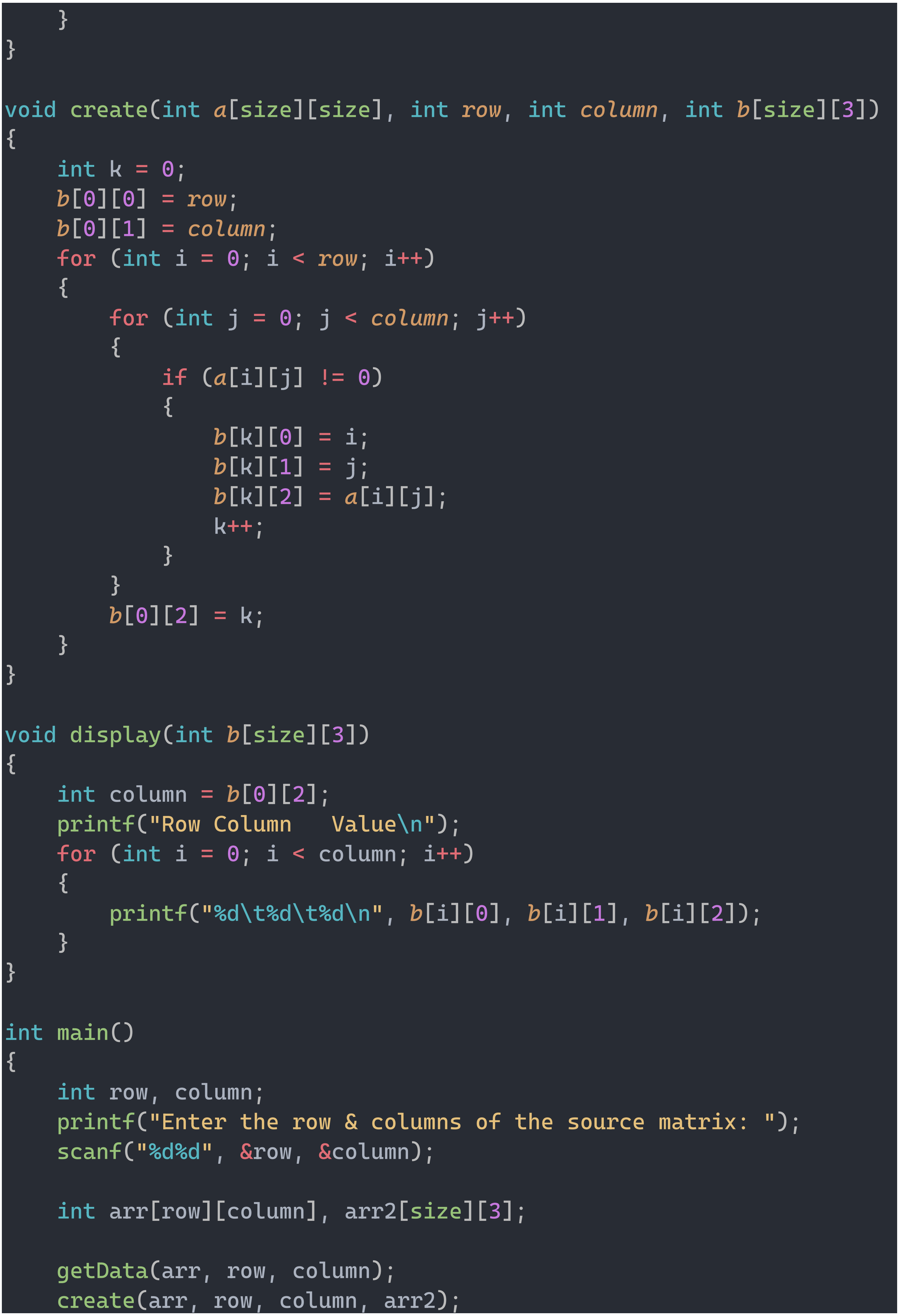


```
 } ; if ($?) { .\q2 }
Creation of Polynomial p
Enter how many terms: 2
Enter the coefficient and exponent of polynomial x = 2 4
Enter the coefficient and exponent of polynomial x = 5 0
Creation of Polynomial q
Enter how many terms: 3
Enter the coefficient and exponent of polynomial x = 3 4
Enter the coefficient and exponent of polynomial x = 6 8
Enter the coefficient and exponent of polynomial x = 2 0
The polynomial p = 2x^4 + 5x^0
The polynomial q = 3x^4 + 6x^8 + 2x^0
The polynomial p + q is r = 5x^4 + 7x^0 + 6x^8
PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DSA LAB
```

Q3 A matrix m × n that has relatively few non-zero entries is called sparse matrix. It may be represented in much less than m × n space. An m × n matrix with k non-zero entries is sparse if k << m × n. It may be faster to represent the matrix compactly as a list of the non-zero indexes and associated entries. WAP to represent a sparse matrix using linked list.

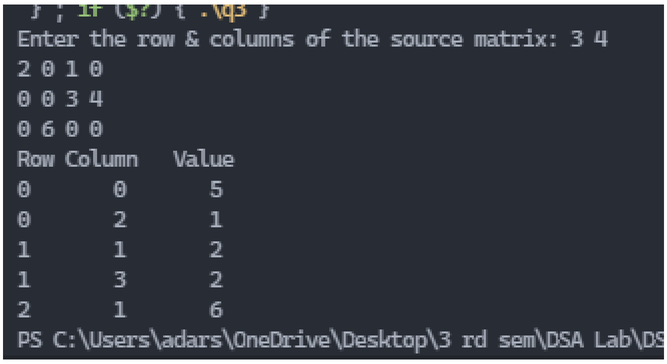


```
#include <stdio.h>
#define size 20

void getData(int a[size][size], int row, int column)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            scanf("%d", &a[i][j]);
        }
```



```c
    }
}

void create(int a[size][size], int row, int column, int b[size][3])
{
    int k = 0;
    b[0][0] = row;
    b[0][1] = column;
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < column; j++)
        {
            if (a[i][j] != 0)
            {
                b[k][0] = i;
                b[k][1] = j;
                b[k][2] = a[i][j];
                k++;
            }
        }
        b[0][2] = k;
    }
}

void display(int b[size][3])
{
    int column = b[0][2];
    printf("Row Column   Value\n");
    for (int i = 0; i < column; i++)
    {
        printf("%d\t%d\t%d\n", b[i][0], b[i][1], b[i][2]);
    }
}

int main()
{
    int row, column;
    printf("Enter the row & columns of the source matrix: ");
    scanf("%d%d", &row, &column);

    int arr[row][column], arr2[size][3];

    getData(arr, row, column);
    create(arr, row, column, arr2);
    display(arr2);

    return 0;
}
```

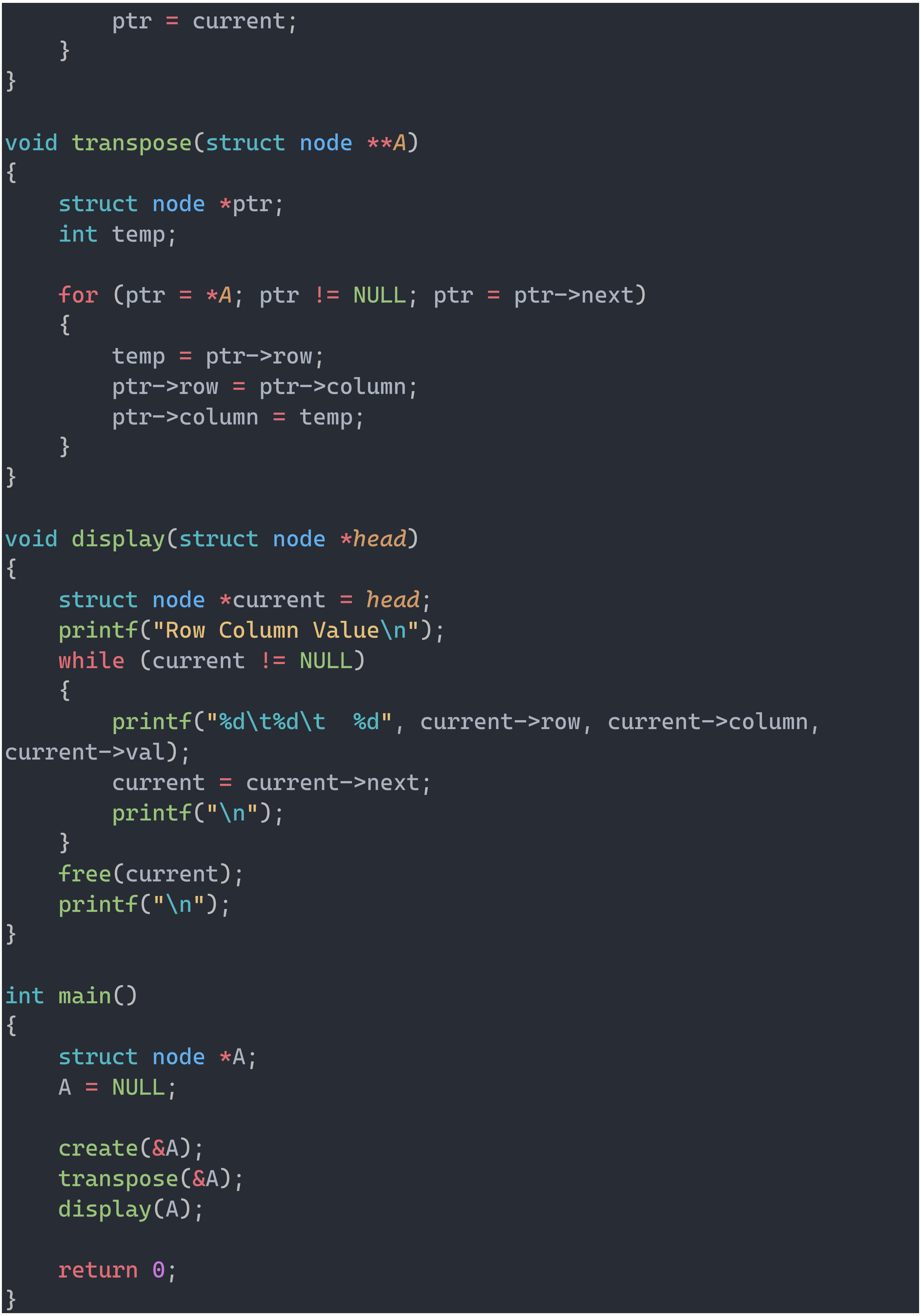
OUTPUT:-



```
 } ; if ($?) { .\q3 }
Enter the row & columns of the source matrix: 3 4
2 0 1 0
0 0 3 4
0 6 0 0
Row Column   Value
0       0       5
0       2       1
1       1       2
1       3       2
2       1       6
PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DS
```

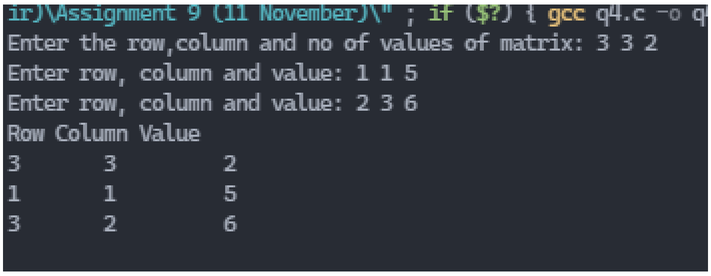
Q4. WAP to find out the transpose of a sparse matrix.

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

struct node
{
    int row;
    int column;
    int val;
    struct node *next;
};

void create(struct node **head)
{
    struct node *ptr, *current;
    *head = malloc(sizeof(struct node));
    (*head)->next = NULL;
    printf("Enter the row,column and no of values of matrix: ");
    scanf("%d%d%d", &(*head)->row, &(*head)->column, &(*head)->val);
    ptr = *head;
    for (int i = 0; i < (*head)->val; i++)
    {
        current = malloc(sizeof(struct node));
        current->next = NULL;
        printf("Enter row, column and value: ");
        scanf("%d%d%d", &current->row, &current->column, &current-
>val);
        ptr->next = current;
```



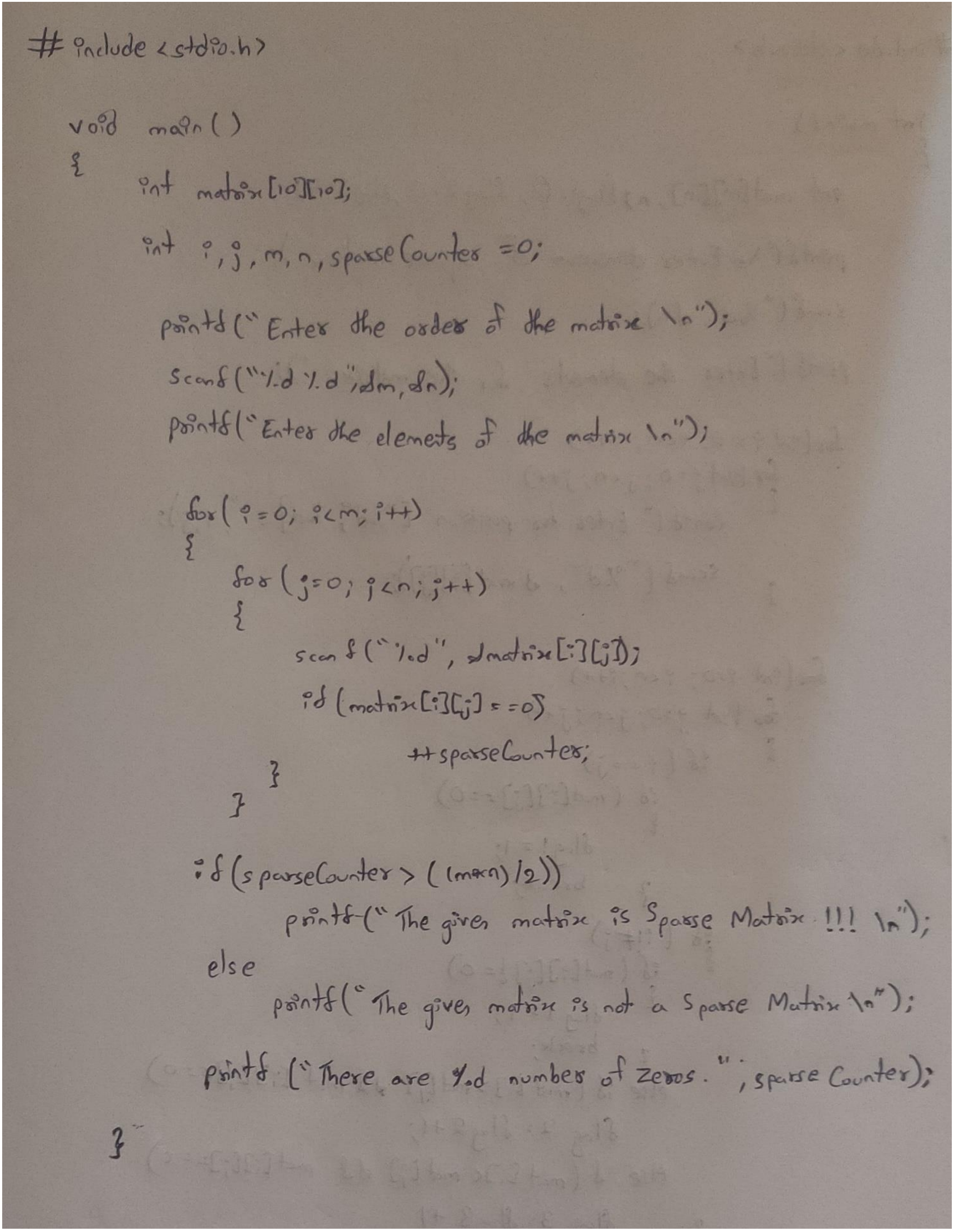
```c
        ptr = current;
    }
}

void transpose(struct node **A)
{
    struct node *ptr;
    int temp;

    for (ptr = *A; ptr != NULL; ptr = ptr->next)
    {
        temp = ptr->row;
        ptr->row = ptr->column;
        ptr->column = temp;
    }
}

void display(struct node *head)
{
    struct node *current = head;
    printf("Row Column Value\n");
    while (current != NULL)
    {
        printf("%d\t%d\t  %d", current->row, current->column,
current->val);
        current = current->next;
        printf("\n");
    }
    free(current);
    printf("\n");
}

int main()
{
    struct node *A;
    A = NULL;

    create(&A);
    transpose(&A);
    display(A);

    return 0;
}
```

OUTPUT:-



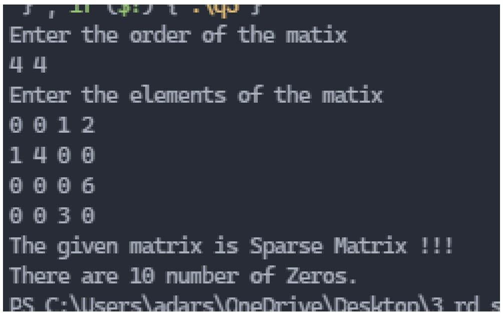
```
ir)\Assignment 9 (11 November)\" ; if ($?) { gcc q4.c -o q
Enter the row,column and no of values of matrix: 3 3 2
Enter row, column and value: 1 1 5
Enter row, column and value: 2 3 6
Row Column Value
3       3         2
1       1         5
3       2         6
```

Q5. WAP to determine whether the given matrix is a sparse matrix or not.



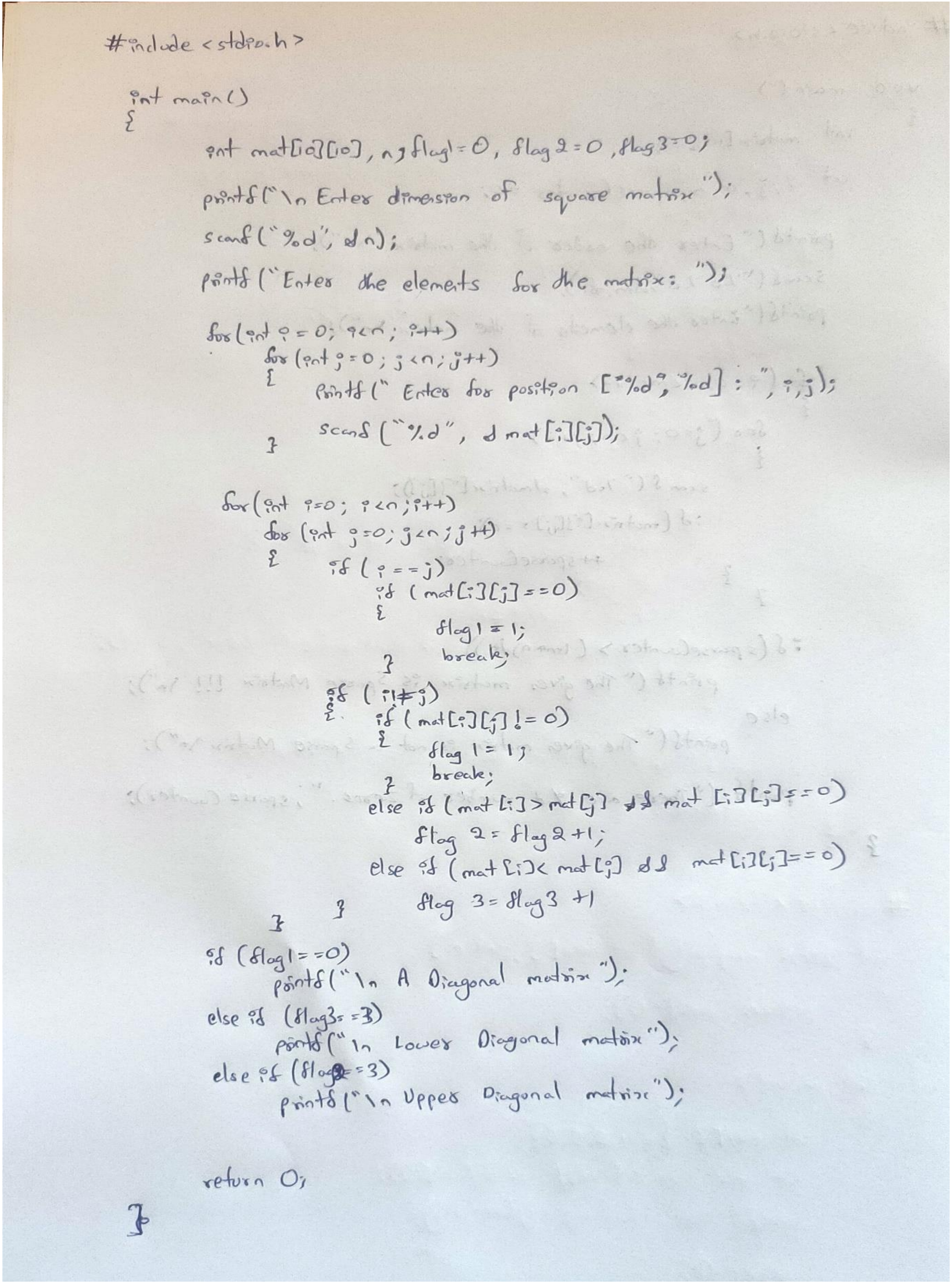
```
# include <stdio.h>

void main ( )
{
    int matrix[10][10];
    int i,j,m,n,sparseCounter =0;

    printf("Enter the order of the matrix \n");
    scanf("%d %d",&m,&n);
    printf("Enter the elemets of the matrix \n");

    for( i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
        {
            scanf("%d", &matrix[i][j]);
            if (matrix[i][j] = =0)
                    ++sparseCounter;
        }
    }
    if(sparseCounter > ((m*n)/2))
        printf("The given matrix is Sparse Matrix !!! \n");
    else
        printf("The given matrix is not a Sparse Matrix \n");

    printf ("There are %d number of Zeros.", sparseCounter);

}
```

OUTPUT:-



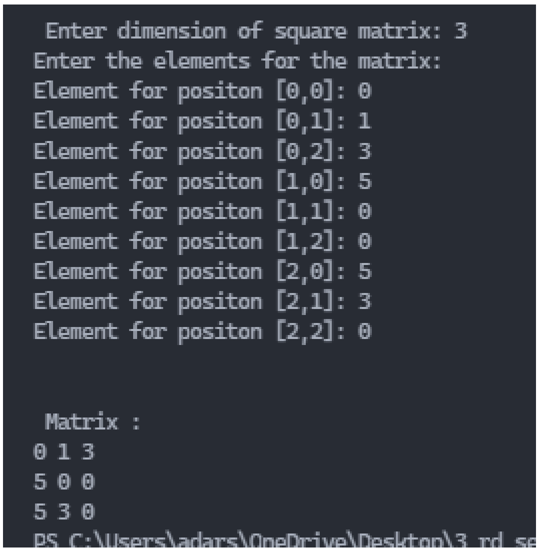
```
} ; if ($?) { .\q5 }
Enter the order of the matix
4 4
Enter the elements of the matix
0 0 1 2
1 4 0 0
0 0 0 6
0 0 3 0
The given matrix is Sparse Matrix !!!
There are 10 number of Zeros.
PS C:\Users\adars\OneDrive\Desktop\3 rd s
```

Q6. WAP to determine whether the given matrix is a lower triangular or upper triangular or tri-diagonal matrix.



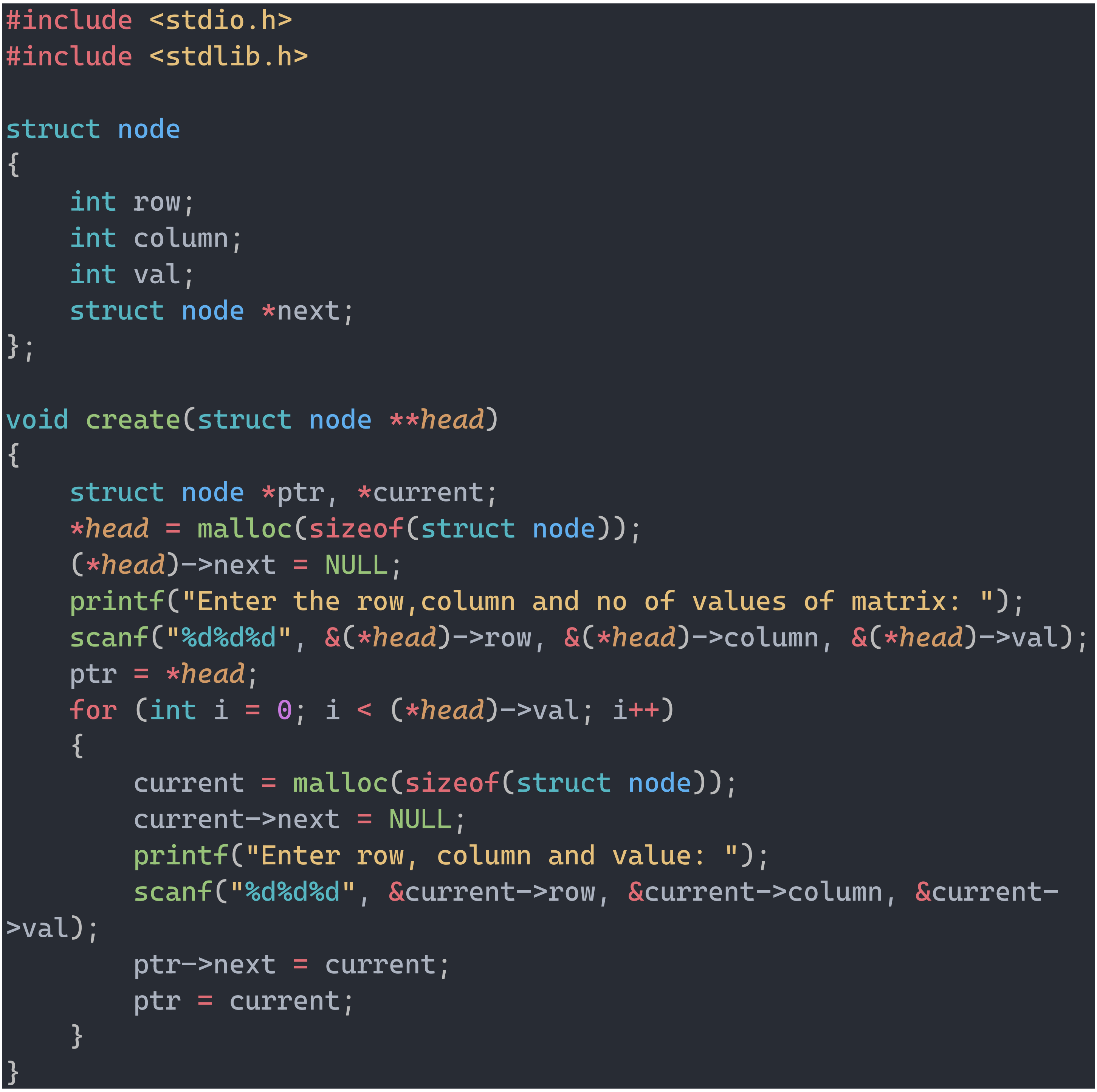
```
#include <stdio.h>

 int main()
 {
        int mat[10][10], n, flag1=0, flag2=0, flag3=0;
        printf("\n Enter dimension of square matrix ");
        scanf("%d", &n);
        printf ("Enter the elements for the matrix: ");

        for(int i = 0; i<n; i++)
            for (int j = 0; j<n; j++)
            {
                 printf (" Enter for position [%d, %d] : ", i, j);
                 scanf ("%d", & mat[i][j]);
            }

        for(int i=0; i<n; i++)
            for (int j=0; j<n; j++)
            {     if ( i == j)
                      if ( mat[i][j]==0)
                      {
                            flag1 = 1;
                            break;
                      }
                  if ( i!=j)
                  {   if ( mat[i][j] != 0)
                      {    flag 1 = 1;
                           break;
                      }
                      else if (mat[i] > mat[j] && mat [i][j]==0)
                           flag 2 = flag2 +1;
                      else if (mat[i]< mat[j] && mat[i][j]==0)
                  }        flag 3 = flag3 +1
            }
        if (flag1==0)
            printf("\n A Diagonal matrix ");
        else if (flag3==3)
            printf("\n Lower Diagonal matrix");
        else if (flag2==3)
            printf("\n Upper Diagonal matrix");

        return 0;
 }
```

OUTPUT:-

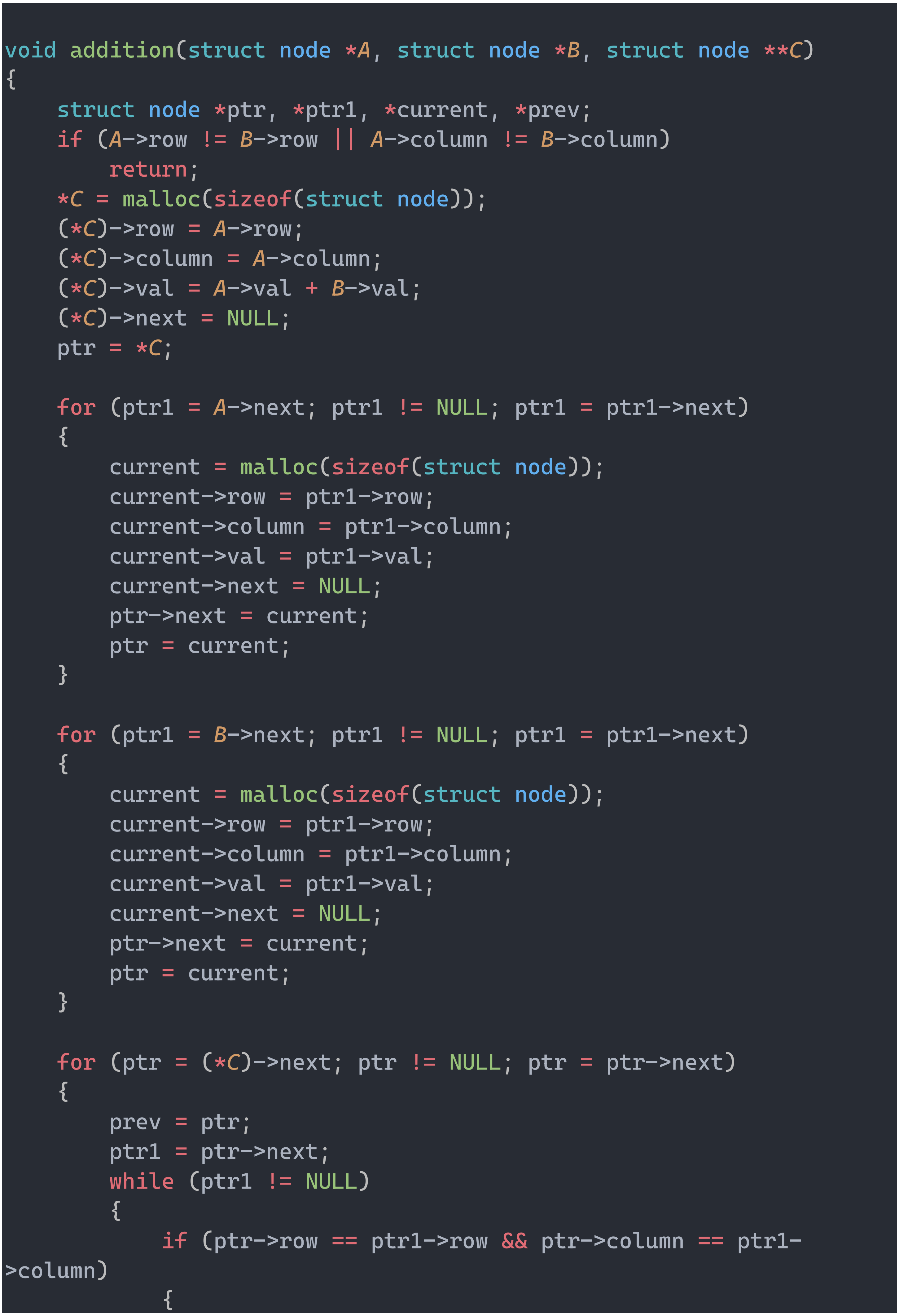


```
 Enter dimension of square matrix: 3
Enter the elements for the matrix:
Element for positon [0,0]: 0
Element for positon [0,1]: 1
Element for positon [0,2]: 3
Element for positon [1,0]: 5
Element for positon [1,1]: 0
Element for positon [1,2]: 0
Element for positon [2,0]: 5
Element for positon [2,1]: 3
Element for positon [2,2]: 0


 Matrix :
0 1 3
5 0 0
5 3 0
PS C:\Users\adars\OneDrive\Desktop\3 rd se
```

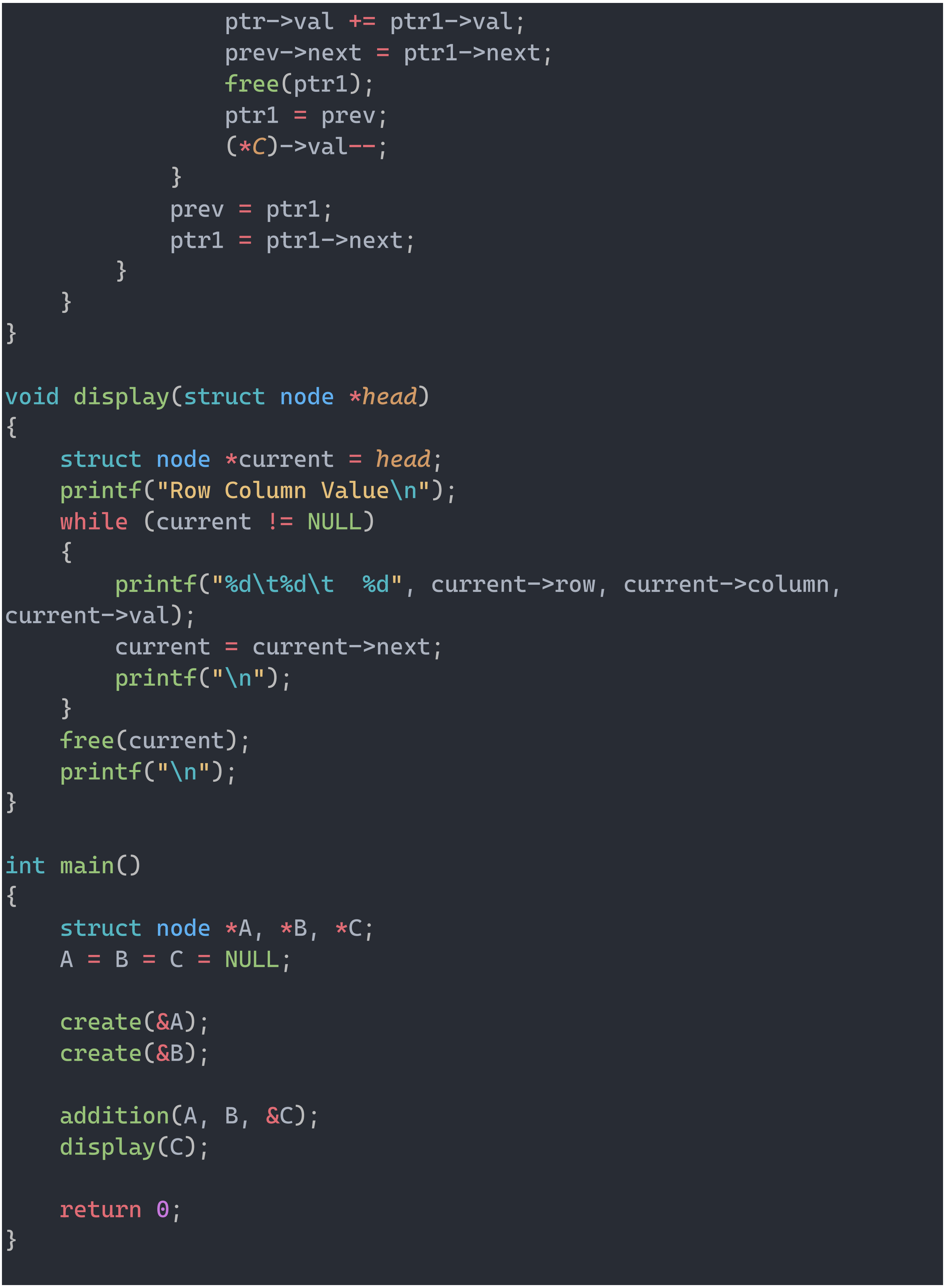
Q7. WAP to add two sparse matrixes.



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

struct node
{
    int row;
    int column;
    int val;
    struct node *next;
};

void create(struct node **head)
{
    struct node *ptr, *current;
    *head = malloc(sizeof(struct node));
    (*head)->next = NULL;
    printf("Enter the row,column and no of values of matrix: ");
    scanf("%d%d%d", &(*head)->row, &(*head)->column, &(*head)->val);
    ptr = *head;
    for (int i = 0; i < (*head)->val; i++)
    {
        current = malloc(sizeof(struct node));
        current->next = NULL;
        printf("Enter row, column and value: ");
        scanf("%d%d%d", &current->row, &current->column, &current-
>val);
        ptr->next = current;
        ptr = current;
    }
}
```

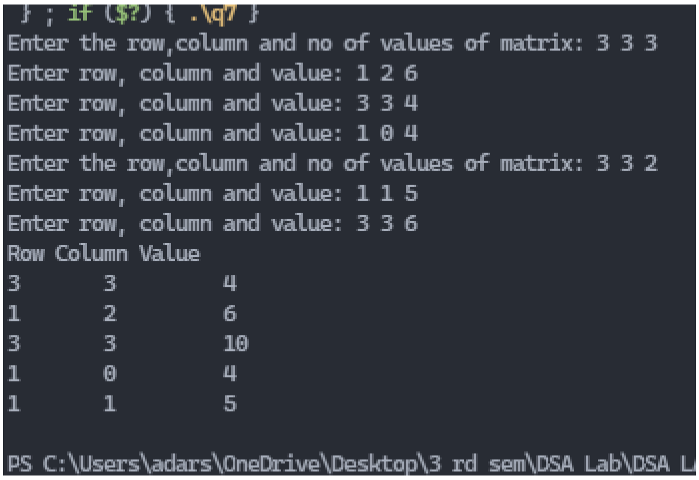


```
void addition(struct node *A, struct node *B, struct node **C)
{
    struct node *ptr, *ptr1, *current, *prev;
    if (A->row != B->row || A->column != B->column)
        return;
    *C = malloc(sizeof(struct node));
    (*C)->row = A->row;
    (*C)->column = A->column;
    (*C)->val = A->val + B->val;
    (*C)->next = NULL;
    ptr = *C;

    for (ptr1 = A->next; ptr1 != NULL; ptr1 = ptr1->next)
    {
        current = malloc(sizeof(struct node));
        current->row = ptr1->row;
        current->column = ptr1->column;
        current->val = ptr1->val;
        current->next = NULL;
        ptr->next = current;
        ptr = current;
    }

    for (ptr1 = B->next; ptr1 != NULL; ptr1 = ptr1->next)
    {
        current = malloc(sizeof(struct node));
        current->row = ptr1->row;
        current->column = ptr1->column;
        current->val = ptr1->val;
        current->next = NULL;
        ptr->next = current;
        ptr = current;
    }

    for (ptr = (*C)->next; ptr != NULL; ptr = ptr->next)
    {
        prev = ptr;
        ptr1 = ptr->next;
        while (ptr1 != NULL)
        {
            if (ptr->row == ptr1->row && ptr->column == ptr1-
>column)
            {
```



```
                ptr->val += ptr1->val;
                prev->next = ptr1->next;
                free(ptr1);
                ptr1 = prev;
                (*C)->val--;
            }
            prev = ptr1;
            ptr1 = ptr1->next;
        }
    }
}

void display(struct node *head)
{
    struct node *current = head;
    printf("Row Column Value\n");
    while (current != NULL)
    {
        printf("%d\t%d\t  %d", current->row, current->column,
current->val);
        current = current->next;
        printf("\n");
    }
    free(current);
    printf("\n");
}

int main()
{
    struct node *A, *B, *C;
    A = B = C = NULL;

    create(&A);
    create(&B);

    addition(A, B, &C);
    display(C);

    return 0;
}
```

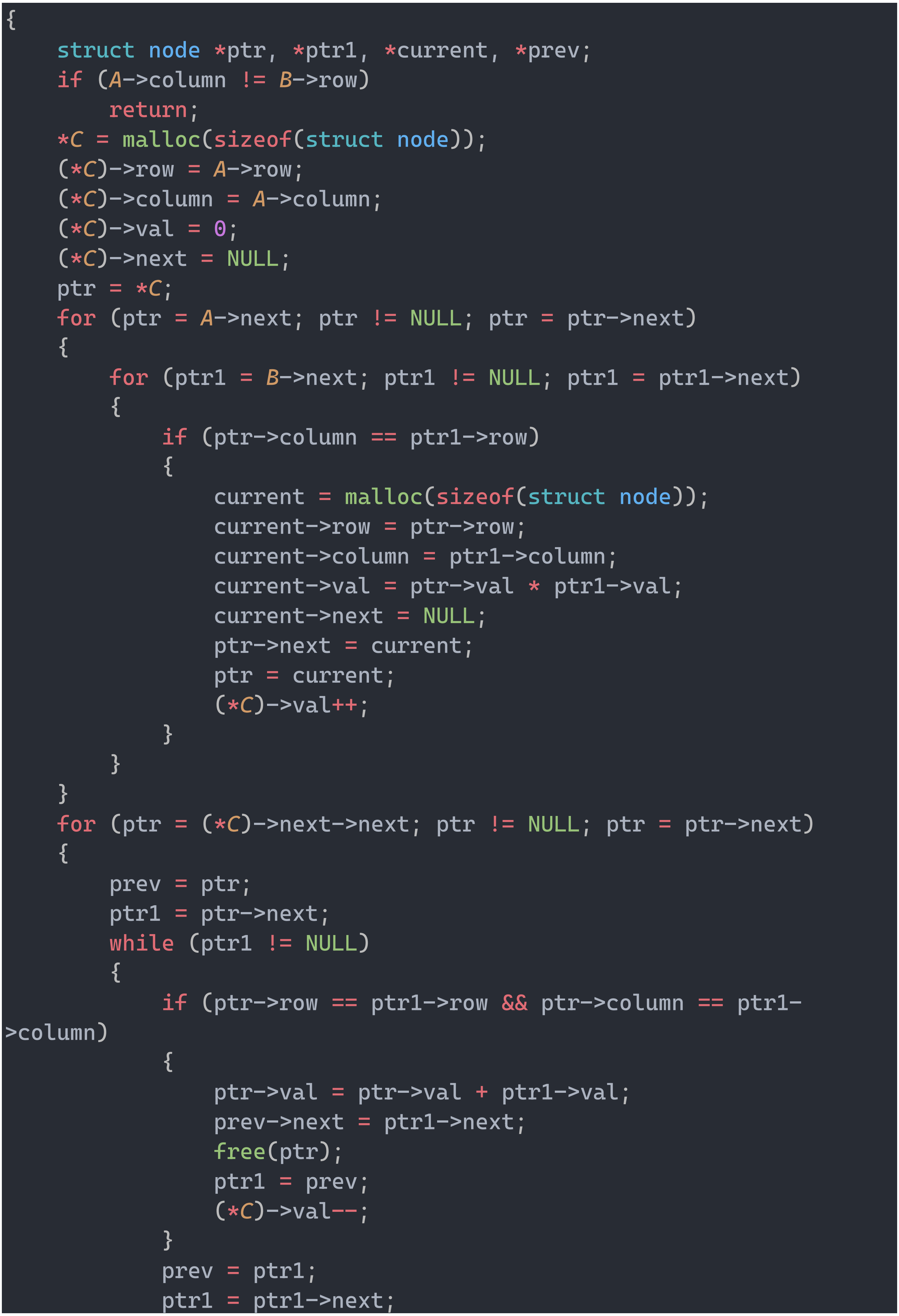
OUTPUT:-



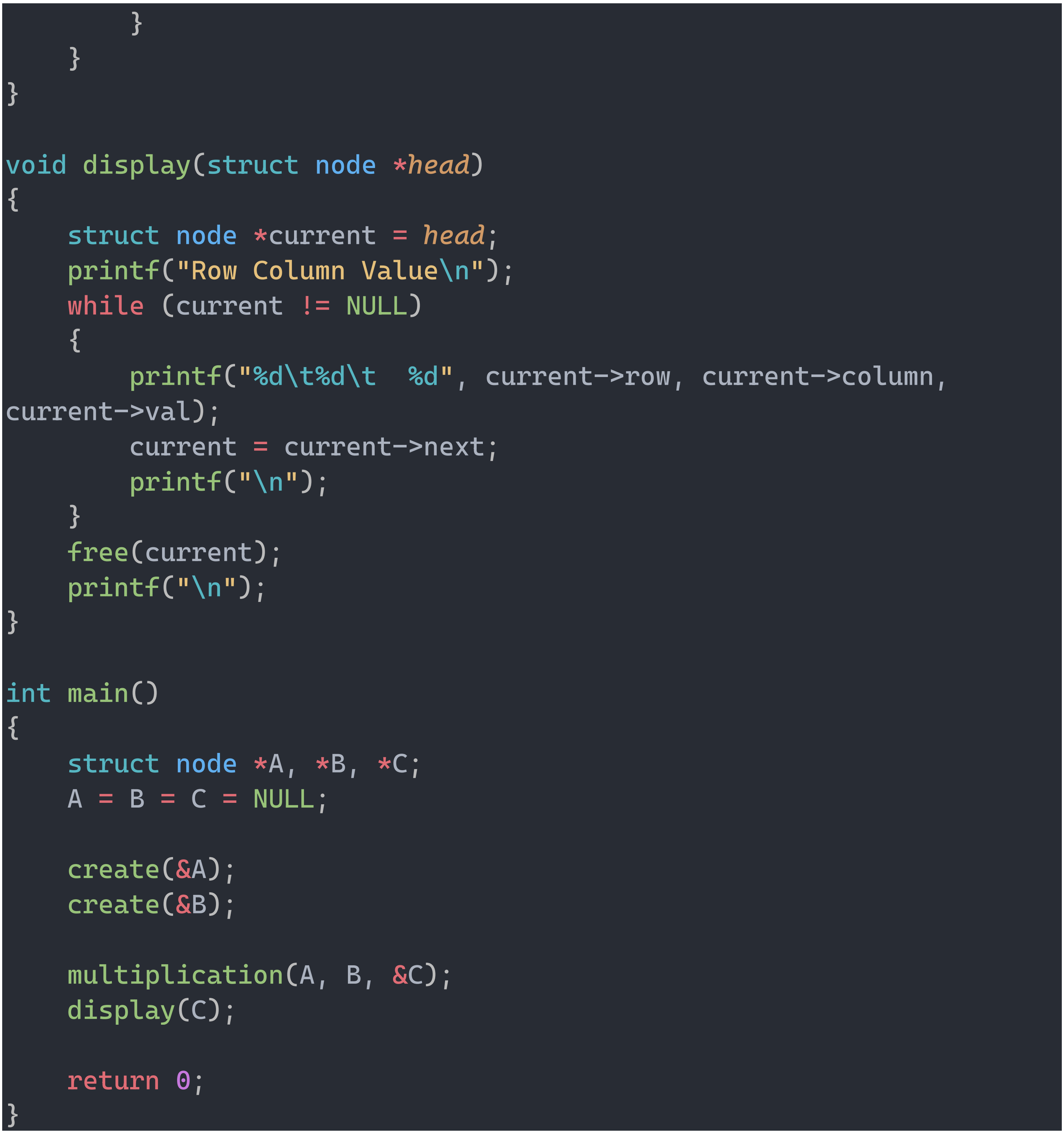
```
 } ; if ($?) { .\q7 }
Enter the row,column and no of values of matrix: 3 3 3
Enter row, column and value: 1 2 6
Enter row, column and value: 3 3 4
Enter row, column and value: 1 0 4
Enter the row,column and no of values of matrix: 3 3 2
Enter row, column and value: 1 1 5
Enter row, column and value: 3 3 6
Row Column Value
3      3        4
1      2        6
3      3        10
1      0        4
1      1        5

PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DSA L
```

Q8. WAP to multiply two sparse matrixes.

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

struct node
{
    int row;
    int column;
    int val;
    struct node *next;
};

void create(struct node **head)
{
    struct node *ptr, *current;
    *head = malloc(sizeof(struct node));
    (*head)->next = NULL;
    printf("Enter the row,column and no of values of matrix: ");
    scanf("%d%d%d", &(*head)->row, &(*head)->column, &(*head)->val);
    ptr = *head;
    for (int i = 0; i < (*head)->val; i++)
    {
        current = malloc(sizeof(struct node));
        current->next = NULL;
        printf("Enter row, column and value: ");
        scanf("%d%d%d", &current->row, &current->column, &current->val);
        ptr->next = current;
        ptr = current;
    }
}

void multiplication(struct node *A, struct node *B, struct node **C)
```

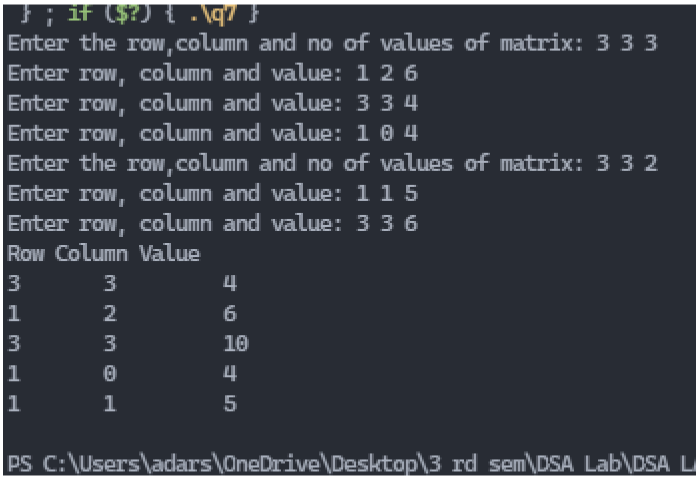


```
{
    struct node *ptr, *ptr1, *current, *prev;
    if (A->column != B->row)
        return;
    *C = malloc(sizeof(struct node));
    (*C)->row = A->row;
    (*C)->column = A->column;
    (*C)->val = 0;
    (*C)->next = NULL;
    ptr = *C;
    for (ptr = A->next; ptr != NULL; ptr = ptr->next)
    {
        for (ptr1 = B->next; ptr1 != NULL; ptr1 = ptr1->next)
        {
            if (ptr->column == ptr1->row)
            {
                current = malloc(sizeof(struct node));
                current->row = ptr->row;
                current->column = ptr1->column;
                current->val = ptr->val * ptr1->val;
                current->next = NULL;
                ptr->next = current;
                ptr = current;
                (*C)->val++;
            }
        }
    }
    for (ptr = (*C)->next->next; ptr != NULL; ptr = ptr->next)
    {
        prev = ptr;
        ptr1 = ptr->next;
        while (ptr1 != NULL)
        {
            if (ptr->row == ptr1->row && ptr->column == ptr1-
>column)
            {
                ptr->val = ptr->val + ptr1->val;
                prev->next = ptr1->next;
                free(ptr);
                ptr1 = prev;
                (*C)->val--;
            }
            prev = ptr1;
            ptr1 = ptr1->next;
```



```c
        }
    }
}

void display(struct node *head)
{
    struct node *current = head;
    printf("Row Column Value\n");
    while (current != NULL)
    {
        printf("%d\t%d\t  %d", current->row, current->column,
current->val);
        current = current->next;
        printf("\n");
    }
    free(current);
    printf("\n");
}

int main()
{
    struct node *A, *B, *C;
    A = B = C = NULL;

    create(&A);
    create(&B);

    multiplication(A, B, &C);
    display(C);

    return 0;
}
```

OUTPUT:-



```
 } ; if ($?) { .\q7 }
Enter the row,column and no of values of matrix: 3 3 3
Enter row, column and value: 1 2 6
Enter row, column and value: 3 3 4
Enter row, column and value: 1 0 4
Enter the row,column and no of values of matrix: 3 3 2
Enter row, column and value: 1 1 5
Enter row, column and value: 3 3 6
Row Column Value
3       3         4
1       2         6
3       3         10
1       0         4
1       1         5

PS C:\Users\adars\OneDrive\Desktop\3 rd sem\DSA Lab\DSA L
```