PROGRAM -01

POLYNOMIAL MULTIPLICATION in “C”

THEORY:

What is a polynomial?

“A polynomial is a sum of terms, where each term has a form ax^e , where x is the variable, a is the coefficient and e is the exponent.”

Two example polynomials are:

A(x) =3x^20 + 2x^5 + 4

B(x) =x^4 + 10x^3 + 3x^2 +1

DESIGN & ALGORITHM:

A simple solution is to one by one consider every term of the first polynomial and multiply it with every term of the second polynomial. Following is the algorithm of this simple method.

```
Multiply (A[0..m-1], B[0..n01])

1) Create a product array prod[ ] of size m+n-1.

2) Initialize all entries in prod[ ] as 0.

3) Traverse array A[ ] and do following for every element A[i]

   Traverse array B[ ] and do following for every element B[j]

        prod[i+j] = prod[i+j] + A[i] * B[j]

4) Return prod[ ].
```

PROGRAM:

```
#include<stdio.h>

#include<stdlib.h>

#define MAX_TERMS 100

typedef struct

{

 int coef;

 int expon;

}polynomial;

polynomial terms[MAX_TERMS];

int avail = 0;

void pmul(int, int, int, int, int *, int * );

void simplifyPoly(int *, int * );

int main()

{

 int startA, startB, finishA, finishB, x, y;

 int *startD, *finishD;

 startD = &x;

 finishD = &y;

 int cf, i, degree1, degree2;

// Reading Polynomial A

 printf("Enter the degree of polynomial a\n");

 scanf("%d", &degree1);

 startA = avail;
 for(; degree1>=0; degree1--)
 {
 printf("Enter the coefficient of %d term of polynomial a\n", degree1);
 scanf("%d", &cf);
 if(cf != 0)
 {
 terms[avail].coef = cf;
 terms[avail++].expon = degree1;
 }
 }
 finishA = avail-1;
 printf("----Polynomial A --------\n");
 for(i=startA; i<finishA; i++)
 printf("%dx^%d + ", terms[i].coef, terms[i].expon);
 printf("%dx^%d = 0\n",terms[i].coef, terms[i].expon);
//Reading Polynomial B
 printf("Enter the degree of polynomial b\n");
 scanf("%d", &degree2);
 startB = avail;
 for(; degree2>=0; degree2--)
 {
 printf("Enter the coefficient of %d term of polynomial a\n", degree2);
 scanf("%d", &cf);
 if(cf)
 {
 terms[avail].coef = cf;
 terms[avail++].expon = degree2;
 }
 }
 finishB = avail-1;
 printf("----Polynomial B --------\n");
 for(i=startB; i<finishB; i++)
 printf("%dx^%d + ", terms[i].coef, terms[i].expon);
 printf("%dx^%d = 0\n",terms[i].coef, terms[i].expon);
// Multiplied polynomial
 pmul(startA, finishA, startB, finishB, startD, finishD);
 printf("----Polynomial D ---------\n");
 for(i=*startD ;i<*finishD; i++)
 printf("%dx^%d + ", terms[i].coef, terms[i].expon);
 printf("%dx^%d = 0\n", terms[i].coef, terms[i].expon);


 return 0;
}
void pmul(int startA, int finishA, int startB, int finishB, int *startD, int
*finishD)
{
 *startD = avail;
 for(int i=startA; i<=finishA; i++)
 {
```

```
 for(int j=startB; j<=finishB; j++)
 {
 terms[avail].coef = terms[i].coef * terms[j].coef;
 terms[avail++].expon = terms[i].expon + terms[j].expon;
 }
 }
 *finishD = avail-1;
 simplifyPoly(startD, finishD);
}
void simplifyPoly(int *startD, int *finishD)
{
 for(int i=*startD; i<=*finishD; i++)
 for(int j=i+1; j<=*finishD; j++)
 if(terms[i].expon == terms[j].expon)
 {
 terms[i].coef = terms[i].coef+terms[j].coef;
 for(int k=j; k<=*finishD; k++)
 terms[k] = terms[k+1];
 *finishD = *finishD - 1;
 }
}
```

OUTPUT:

```
PS C:\Users\dell\Desktop\dsa programs> cd "c:\Users\dell\Desktop\dsa
programs\" ; if ($?)

{ gcc poly.c -o poly } ; if ($?) { .\poly }

Enter the degree of polynomial a

3

Enter the coefficient of 3 term of polynomial a

4

Enter the coefficient of 2 term of polynomial a

6

Enter the coefficient of 1 term of polynomial a

7

Enter the coefficient of 0 term of polynomial a

8

----Polynomial A --------

4x^3 + 6x^2 + 7x^1 + 8x^0 = 0

Enter the degree of polynomial b

4

Enter the coefficient of 4 term of polynomial a

5

Enter the coefficient of 3 term of polynomial a

2

Enter the coefficient of 2 term of polynomial a

9

Enter the coefficient of 1 term of polynomial a
```

10

Enter the coefficient of 0 term of polynomial a

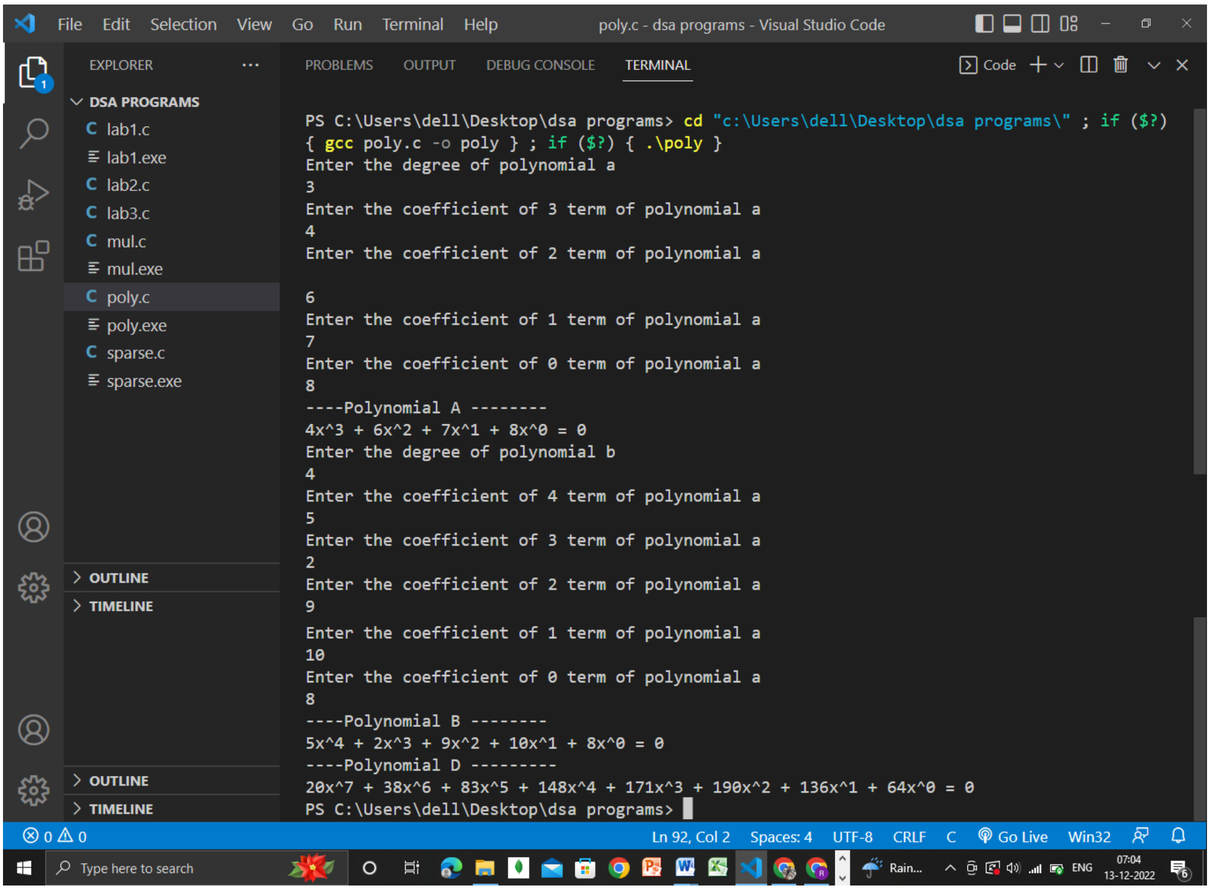
8

----Polynomial B --------

5x^4 + 2x^3 + 9x^2 + 10x^1 + 8x^0 = 0

----Polynomial D ---------

20x^7 + 38x^6 + 83x^5 + 148x^4 + 171x^3 + 190x^2 + 136x^1 + 64x^0 = 0


File Edit Selection View Go Run Terminal Help
poly.c - dsa programs - Visual Studio Code
EXPLORER
DSA PROGRAMS
lab1.c
lab1.exe
lab2.c
lab3.c
mul.c
mul.exe
poly.c
poly.exe
sparse.c
sparse.exe
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
Ln 92, Col 2 Spaces: 4 UTF-8 CRLF C Go Live Win32
Type here to search
Rain...
ENG
07:04
13-12-2022

```
PS C:\Users\dell\Desktop\dsa programs> cd "c:\Users\dell\Desktop\dsa programs\" ; if ($?)
{ gcc poly.c -o poly } ; if ($?) { .\poly }
Enter the degree of polynomial a
3
Enter the coefficient of 3 term of polynomial a
4
Enter the coefficient of 2 term of polynomial a

6
Enter the coefficient of 1 term of polynomial a
7
Enter the coefficient of 0 term of polynomial a
8
----Polynomial A --------
4x^3 + 6x^2 + 7x^1 + 8x^0 = 0
Enter the degree of polynomial b
4
Enter the coefficient of 4 term of polynomial a
5
Enter the coefficient of 3 term of polynomial a
2
Enter the coefficient of 2 term of polynomial a
9
Enter the coefficient of 1 term of polynomial a
10
Enter the coefficient of 0 term of polynomial a
8
----Polynomial B --------
5x^4 + 2x^3 + 9x^2 + 10x^1 + 8x^0 = 0
----Polynomial D ---------
20x^7 + 38x^6 + 83x^5 + 148x^4 + 171x^3 + 190x^2 + 136x^1 + 64x^0 = 0
PS C:\Users\dell\Desktop\dsa programs>
```

Fig 1: Output Screenshot

PROGRAM -02

TRANSPOSE OF A SPARSE MATRIX in “C”

THEORY:

- *What is Sparse Matrix?*

A matrix which contains many zero entries or very few non-zero entries is called as Sparse matrix.

- *Transposing a Matrix*

To transpose a matrix, interchange the rows and columns. This means that each element a[i][j] in the original matrix becomes element a[j][i] in the transpose matrix.

- *Sparse Matrix Representation*

An element within a matrix can characterize by using the triple <row,col,value>, an array of triples is used to represent a sparse matrix. Organize the triples so that the row indices are in ascending order. The operations should terminate, so we must know the number of rows and columns, and the number of nonzero elements in the matrix.

a[0].row contains the number of rows,

a[0].col contains the number of columns

a[0].value contains the total number of nonzero entries.

DESIGN:

To Transpose a matrix, we can simply change every column value to the row value and vice-versa, however, in this case, the resultant matrix won’t be sorted as we require. Hence, we initially determine the number of elements less than the current element’s column being inserted in order to get the exact index of the resultant matrix where the current element should be placed. This is done by maintaining an array index[] whose ith value indicates the number of elements in the matrix less than the column i.

PROGRAM:

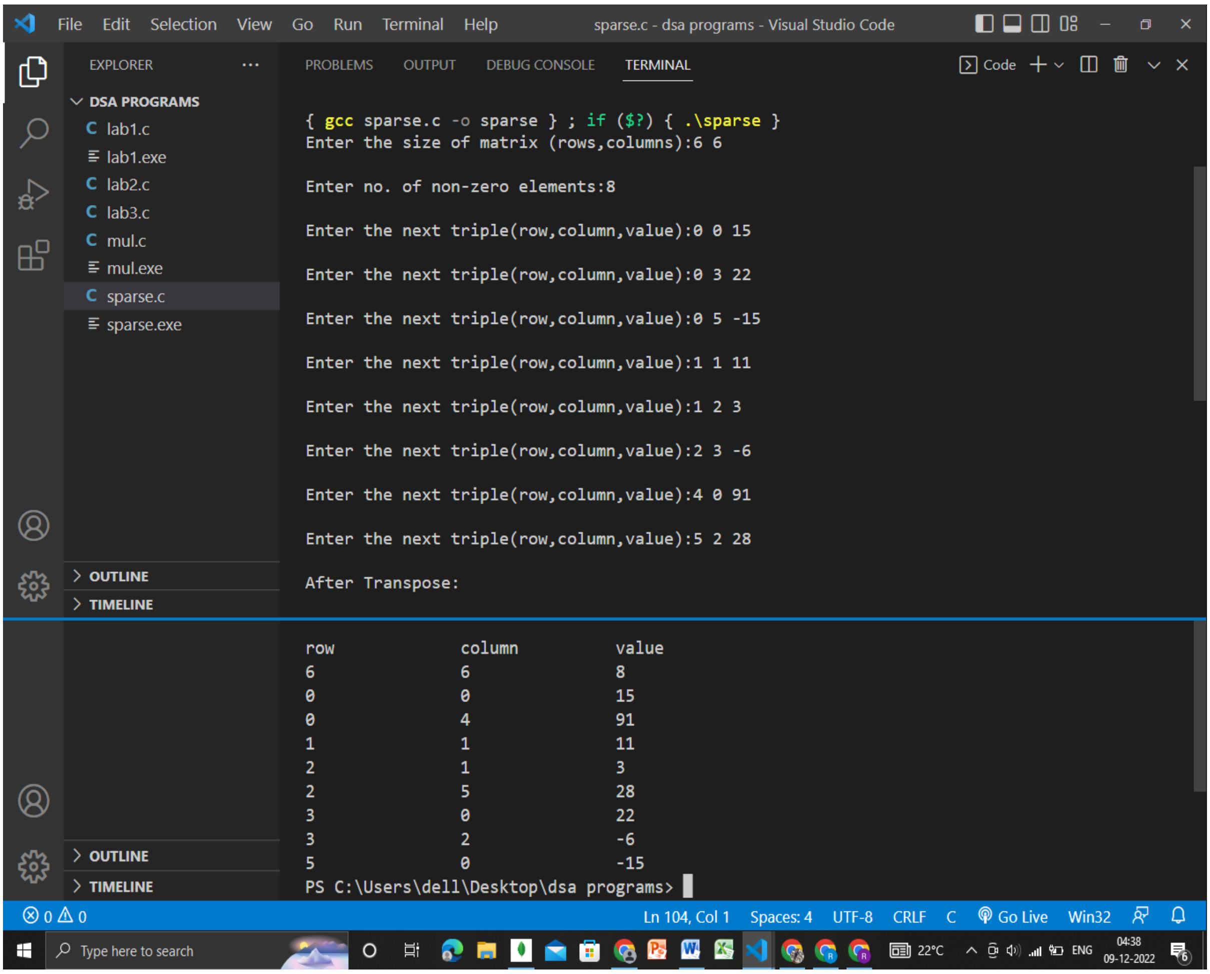
```
/* C program to read sparse matrix and find the transpose of a matrix*/
#include<stdio.h>
#define MAX 20
void printsparse(int[][3]);
void readsparse(int[][3]);
void transpose(int[][3],int[][3]);

int main()
{
int b1[MAX][3], b2[MAX][3], m, n;
printf("Enter the size of matrix (rows,columns):");
scanf("%d%d",&m,&n);
b1[0][0]=m;
b1[0][1]=n;
readsparse(b1);
transpose(b1,b2);
printsparse(b2);
}
void readsparse(int b[MAX][3])
{
int i,t;
printf("\nEnter no. of non-zero elements:");
scanf("%d",&t);
b[0][2]=t;
for(i=1;i<=t;i++)
{
```

```
printf("\nEnter the next triple(row,column,value):");
scanf("%d%d%d",&b[i][0],&b[i][1],&b[i][2]);
}
}
void printsparse(int b[MAX][3])
{
int i,n;
n=b[0][2];
//no of 3-triples
printf("\nAfter Transpose:\n");
printf("\nrow\t\tcolumn\t\tvalue\n");
for(i=0;i<=n;i++)
printf("%d\t\t%d\t\t%d\n",b[i][0],b[i][1],b[i][2]);
}
void transpose(int b1[][3],int b2[][3])
{
int i,j,k,n;
b2[0][0]=b1[0][1];
b2[0][1]=b1[0][0];
b2[0][2]=b1[0][2];
k=1;
n=b1[0][2];
for(i=0;i<b1[0][1];i++)
for(j=1;j<=n;j++)
if(i==b1[j][1])
{
```

```
b2[k][0]=b1[j][1];
b2[k][1]=b1[j][0];
b2[k][2]=b1[j][2];
k++;
}
}
```

OUTPUT:

```
PS C:\Users\dell\Desktop\dsa programs> cd "c:\Users\dell\Desktop\dsa programs\" ; if ($?)
{ gcc sparse.c -o sparse } ; if ($?) { .\sparse }
Enter the size of matrix (rows,columns):6 6

Enter no. of non-zero elements:8

Enter the next triple(row,column,value):0 0 15

Enter the next triple(row,column,value):0 3 22

Enter the next triple(row,column,value):0 5 -15

Enter the next triple(row,column,value):1 1 11

Enter the next triple(row,column,value):1 2 3

Enter the next triple(row,column,value):2 3 -6

Enter the next triple(row,column,value):4 0 91
Enter the next triple(row,column,value):5 2 28

After Transpose:

row         column      value

6             6           8

0             0           15

0             4           91

1             1           11

2             1           3

3             0           22

3             2           -6

5             0           -15
```



```
{ gcc sparse.c -o sparse } ; if ($?) { .\sparse }
Enter the size of matrix (rows,columns):6 6

Enter no. of non-zero elements:8

Enter the next triple(row,column,value):0 0 15

Enter the next triple(row,column,value):0 3 22

Enter the next triple(row,column,value):0 5 -15

Enter the next triple(row,column,value):1 1 11

Enter the next triple(row,column,value):1 2 3

Enter the next triple(row,column,value):2 3 -6

Enter the next triple(row,column,value):4 0 91

Enter the next triple(row,column,value):5 2 28

After Transpose:

row             column          value
6               6               8
0               0               15
0               4               91
1               1               11
2               1               3
2               5               28
3               0               22
3               2               -6
5               0               -15
PS C:\Users\dell\Desktop\dsa programs>
```

Fig 2:Output Screenshot