K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent College of Somaiya Vidyavihar University)

Ketaki Mahajan

Batch: P1 - 2 **Roll No.: 16014022050**

Experiment / assignment / tutorial No. 7

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE:
Using virtual labs to understand the concept of matrix multiplication, call by reference

AIM: Use of virtual labs to understand the concepts and theory with examples and verify the same with practice questions.

Expected OUTCOME of Experiment:
Illustrate the use of derived and structured data types such as arrays, strings, structures and unions.

Books/ Journals/ Websites referred:

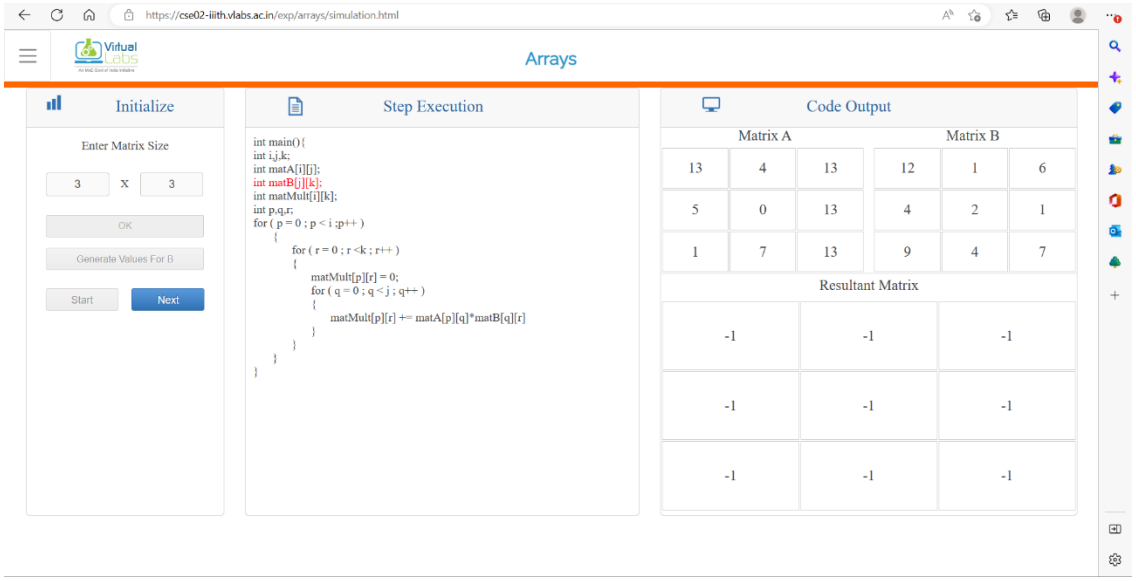
1. Programming in ANSI C, E. Balagurusamy, 7 th Edition, 2016, McGraw-Hill Education, India.
2. Structured Programming Approach, Pradeep Dey and Manas Ghosh, 1 st Edition, 2016, Oxford University Press, India.
3. Let Us C, Yashwant Kanetkar, 15th Edition, 2016, BPB Publications, India.

Problem Definition:
Virtual Lab experiment on matrix multiplication
https://cse02-iiith.vlabs.ac.in/exp/arrays/simulation.html

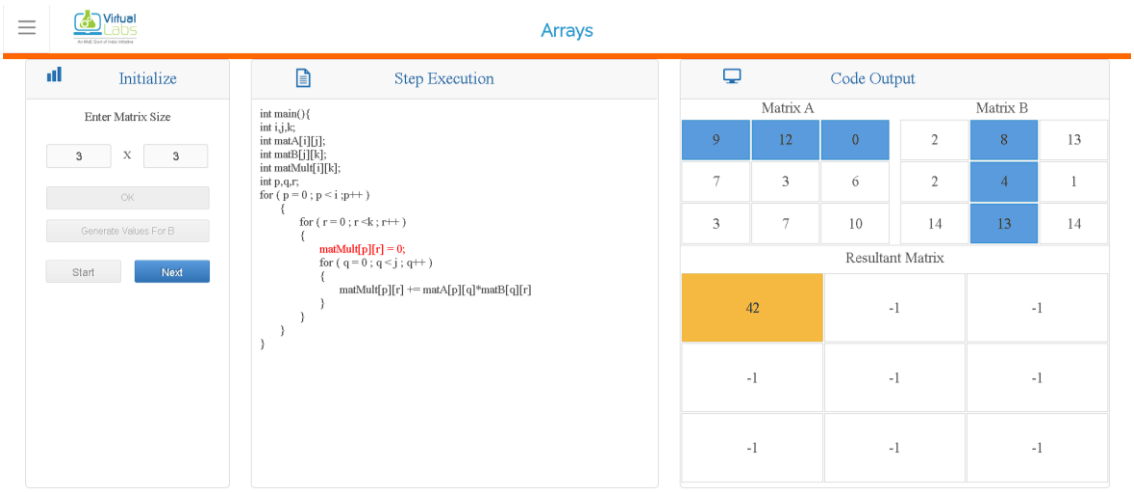
Virtual Lab experiment on Call by reference
https://cse02-iiith.vlabs.ac.in/exp/pointers/procedure.html

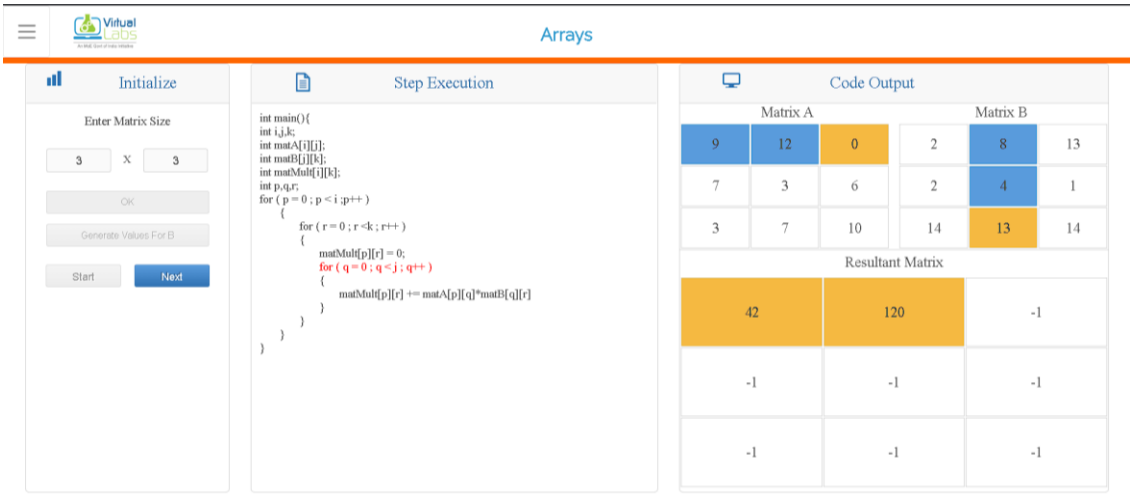
Program to swap two number without using third variable using Call by reference.

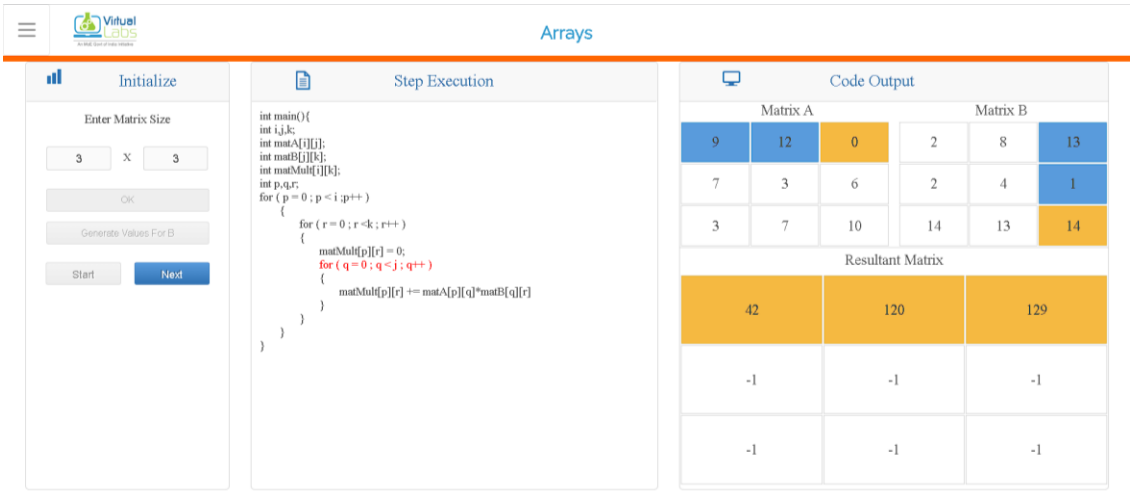
Implementation details/ Simulation screenshots:

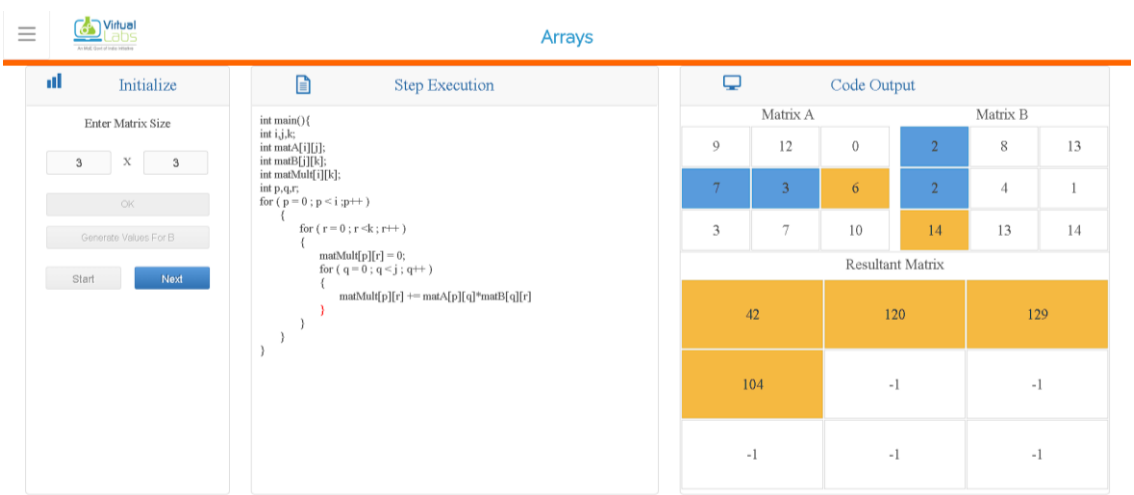


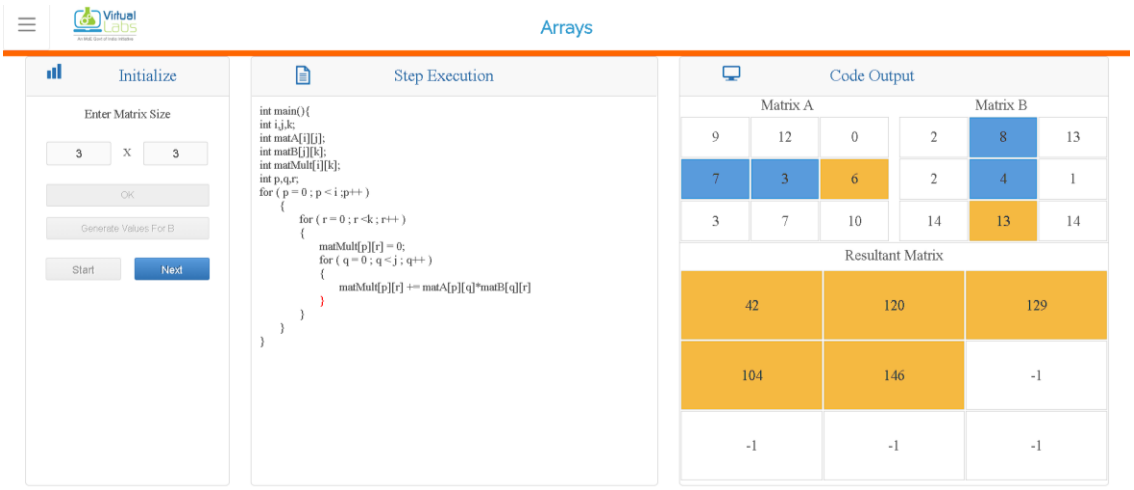
Output(s)/Post-test Screenshots:

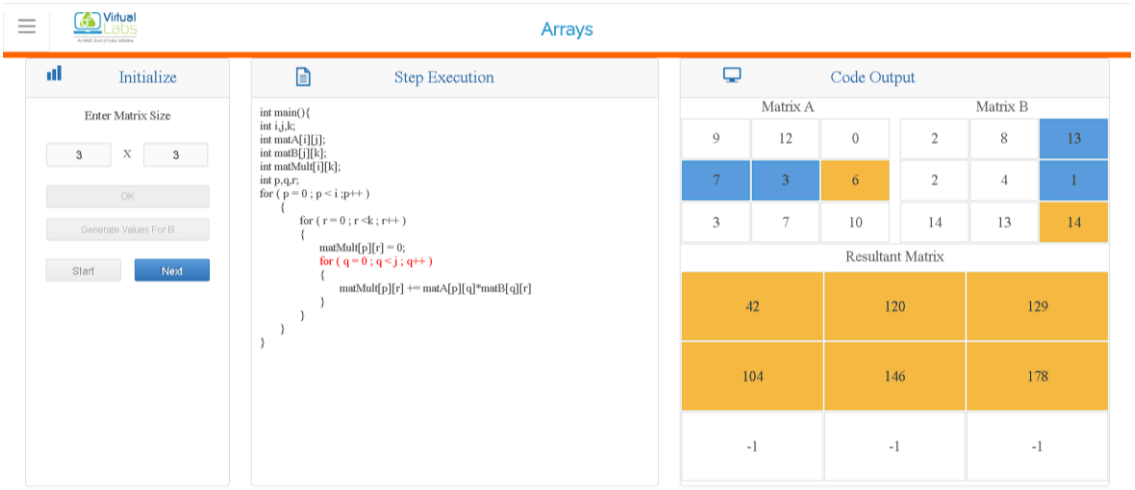


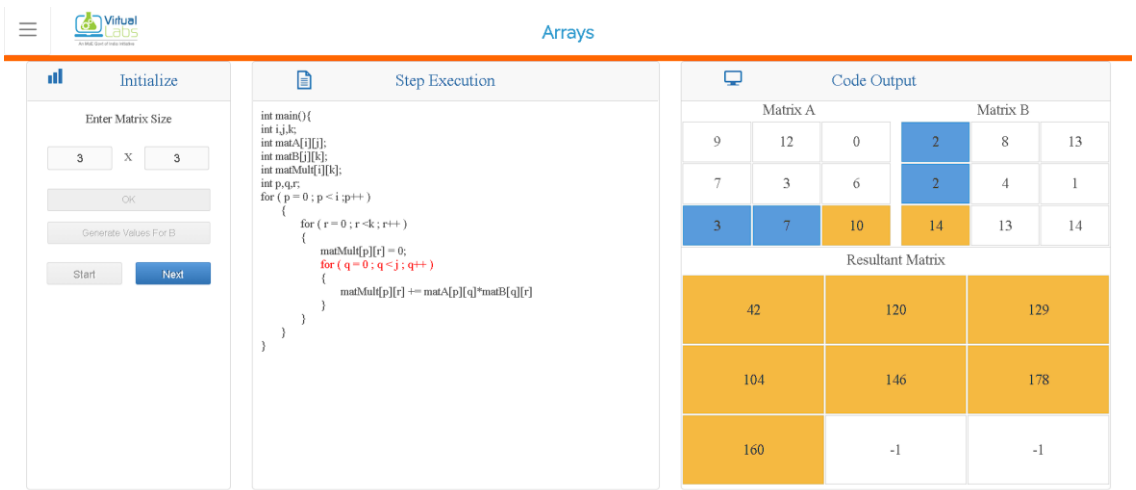

Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q < j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0
7 3 6
3 7 10
2 8 13
2 4 1
14 13 14
Resultant Matrix
42 120 -1
-1 -1 -1
-1 -1 -1

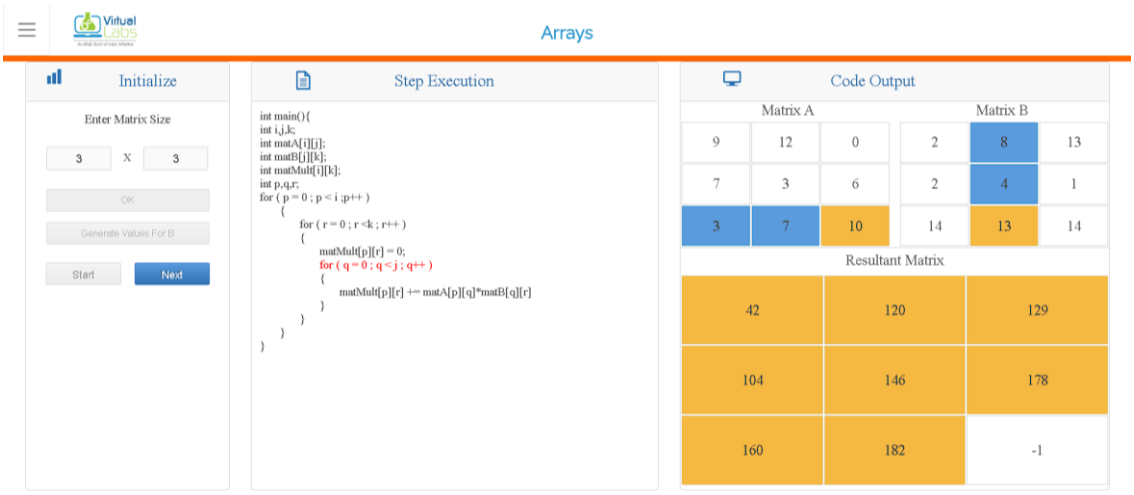

Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q < j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0
7 3 6
3 7 10
2 8 13
4 1
14 13 14
Resultant Matrix
42 120 129
-1 -1 -1
-1 -1 -1

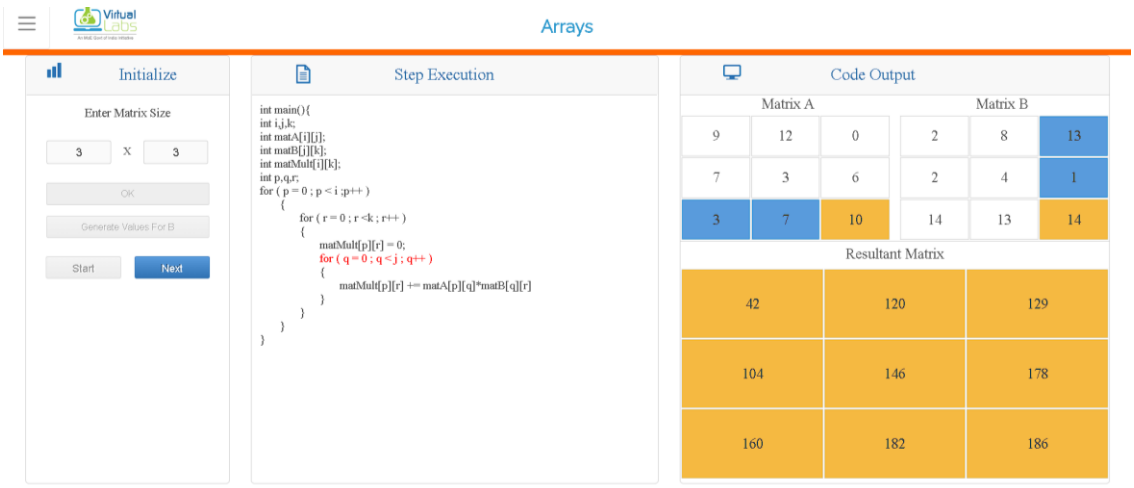

Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q < j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0
7 3 6
3 7 10
2 8 13
2 4 1
14 13 14
Resultant Matrix
42 120 129
104 -1 -1
-1 -1 -1


Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q < j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0 2 8 13
7 3 6 2 4 1
3 7 10 14 13 14
Resultant Matrix
42 120 129
104 146 -1
-1 -1 -1


Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q < j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0 2 8 13
7 3 6 2 4 1
3 7 10 14 13 14
Resultant Matrix
42 120 129
104 146 178
-1 -1 -1


Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q < j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0 2 8 13
7 3 6 2 4 1
3 7 10 14 13 14
Resultant Matrix
42 120 129
104 146 178
160 -1 -1


Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q <j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0 2 8 13
7 3 6 2 4 1
3 7 10 14 13 14
Resultant Matrix
42 120 129
104 146 178
160 182 -1


Arrays
Initialize
Enter Matrix Size
3 X 3
OK
Generate Values For B
Start
Next
Step Execution
int main(){
int i,j,k;
int matA[i][j];
int matB[j][k];
int matMult[i][k];
int p,q,r;
for (p = 0 ; p < i ;p++)
{
for (r = 0 ; r <k ; r++)
{
matMult[p][r] = 0;
for (q = 0 ; q <j ; q++)
{
matMult[p][r] += matA[p][q]*matB[q][r]
}
}
}
}
Code Output
Matrix A
Matrix B
9 12 0 2 8 13
7 3 6 2 4 1
3 7 10 14 13 14
Resultant Matrix
42 120 129
104 146 178
160 182 186

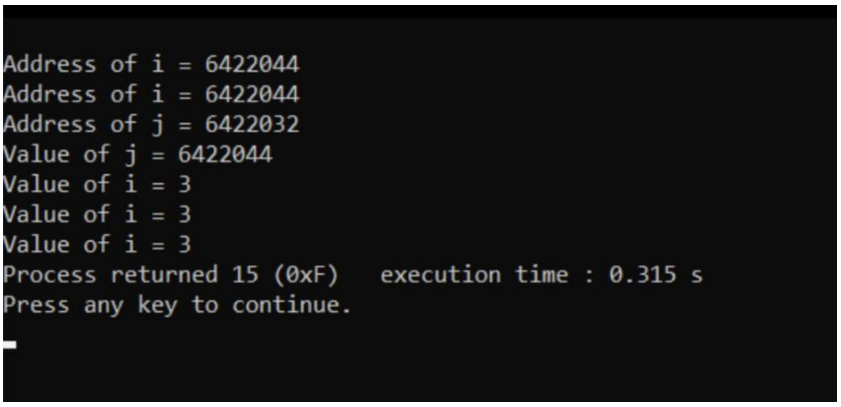
Conclusion and your take away after performing the virtual lab experiment: -
The program to sort 1D array and to multiply two 2D arrays were learnt using online simulation lab. The simulation was successfully completed.

Post Lab Descriptive Questions:

1. Differentiate between Call by Value and Call by Reference.
   Call by Value: In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.
   Call by Reference: Both the actual and formal parameters refer to the same locations, so any changes made inside the function are actually reflected in actual parameters of the caller.

2. Try to understand the working of pointers by Running the following code and noting down the output.

```
main ( )
{
        int i = 3 ;
        int *j ;
        j = &i ;
        printf ( "\nAddress of i = %u", &i ) ;
        printf ( "\nAddress of i = %u", j ) ;
        printf ( "\nAddress of j = %u", &j ) ;
        printf ( "\nValue of j = %u", j ) ;
        printf ( "\nValue of i = %d", i ) ;
        printf ( "\nValue of i = %d", *( &i ) ) ;
        printf ( "\nValue of i = %d", *j ) ;
}
```



Date: _____ **Signature of faculty in-charge**