**Batch: B2 Roll No.: 121**

**Experiment / assignment / tutorial No. 7**

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

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:**  Virtual Lab experiment on matrix multiplication |

**AIM:** Virtual Lab experiment on recursion

<https://cse02-iiith.vlabs.ac.in/>

<https://cse02-iiith.vlabs.ac.in/exp/arrays/simulation.html>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO3: Illustrate the use of derived and structured data types such as arrays, strings, structures and unions.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.
4. [**http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**](http://cse.iitkgp.ac.in/~rkumar/pds-vlab/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

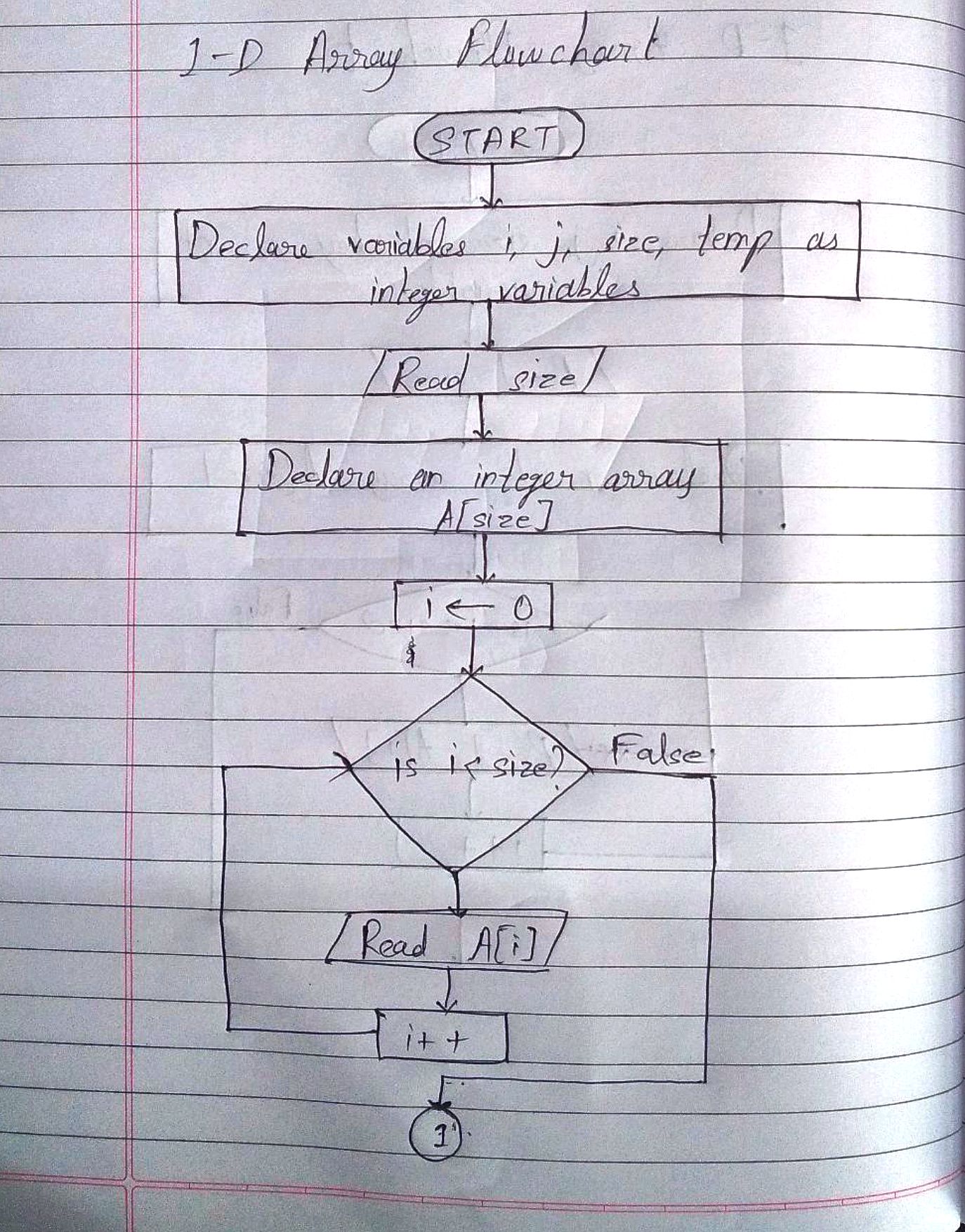
The Program implements 1-D and 2-D arrays.

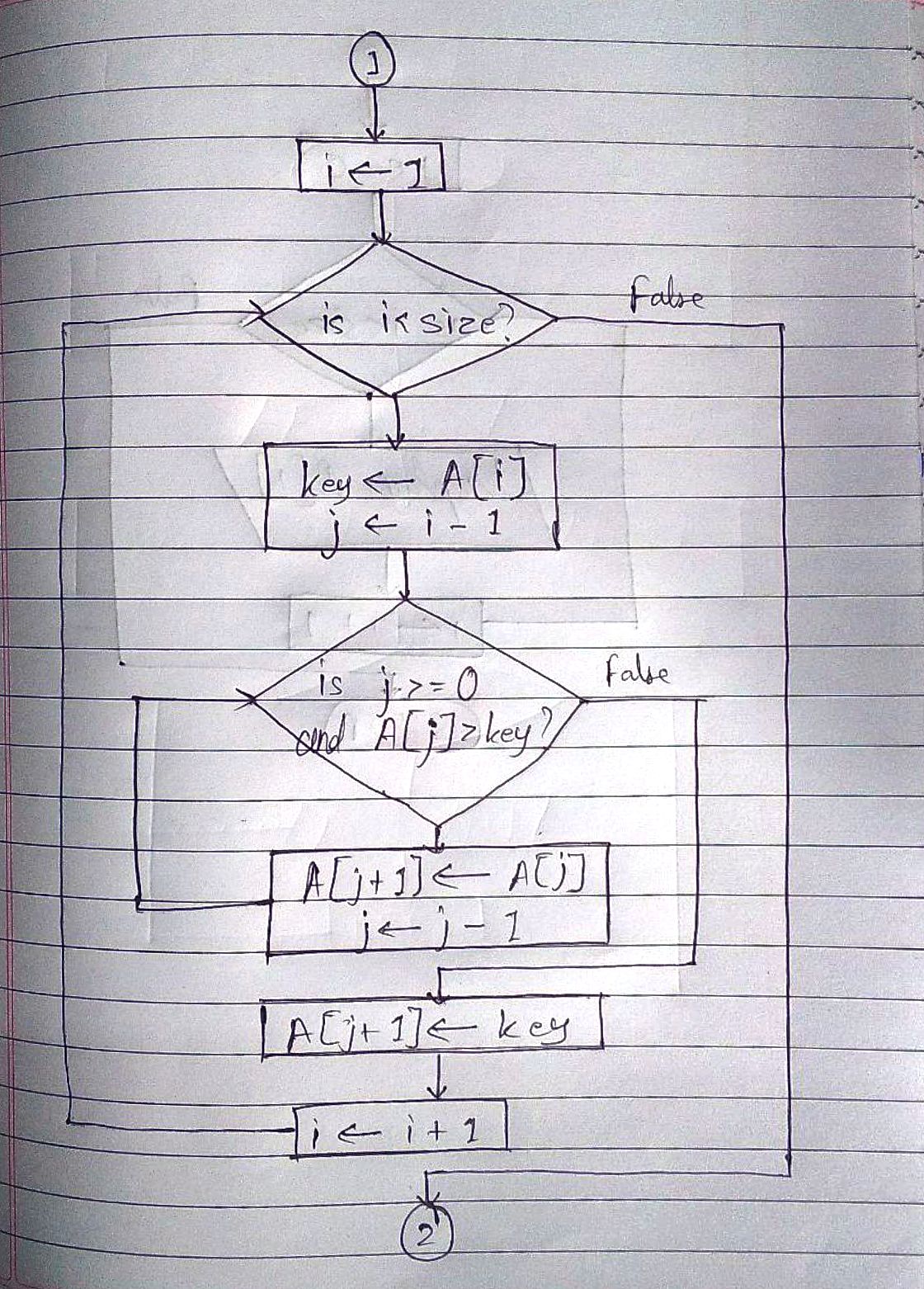
1-D array prompts the user to enter the size of the array and elements. It then sorts the elements.

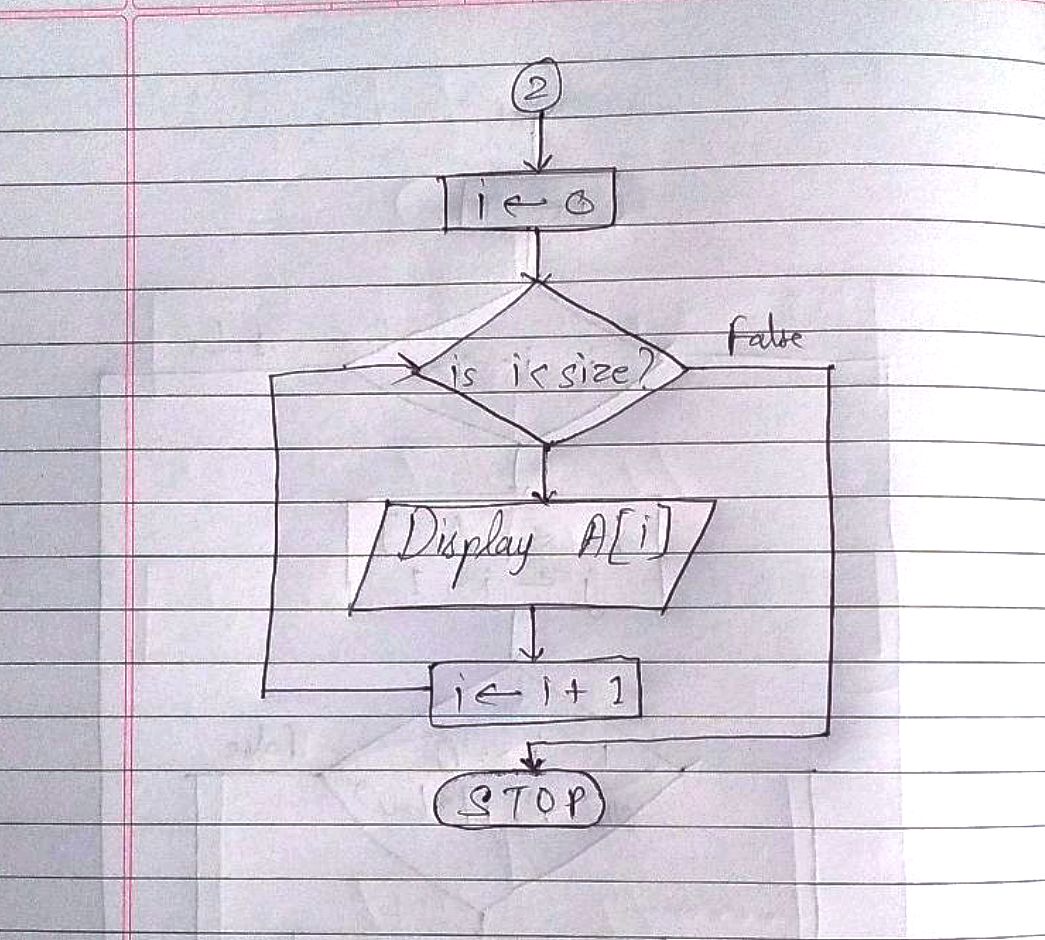
2-D array prompts the user to enter the order of two matrices and the elements. It then performs matrix multiplication.

**Flowchart:**

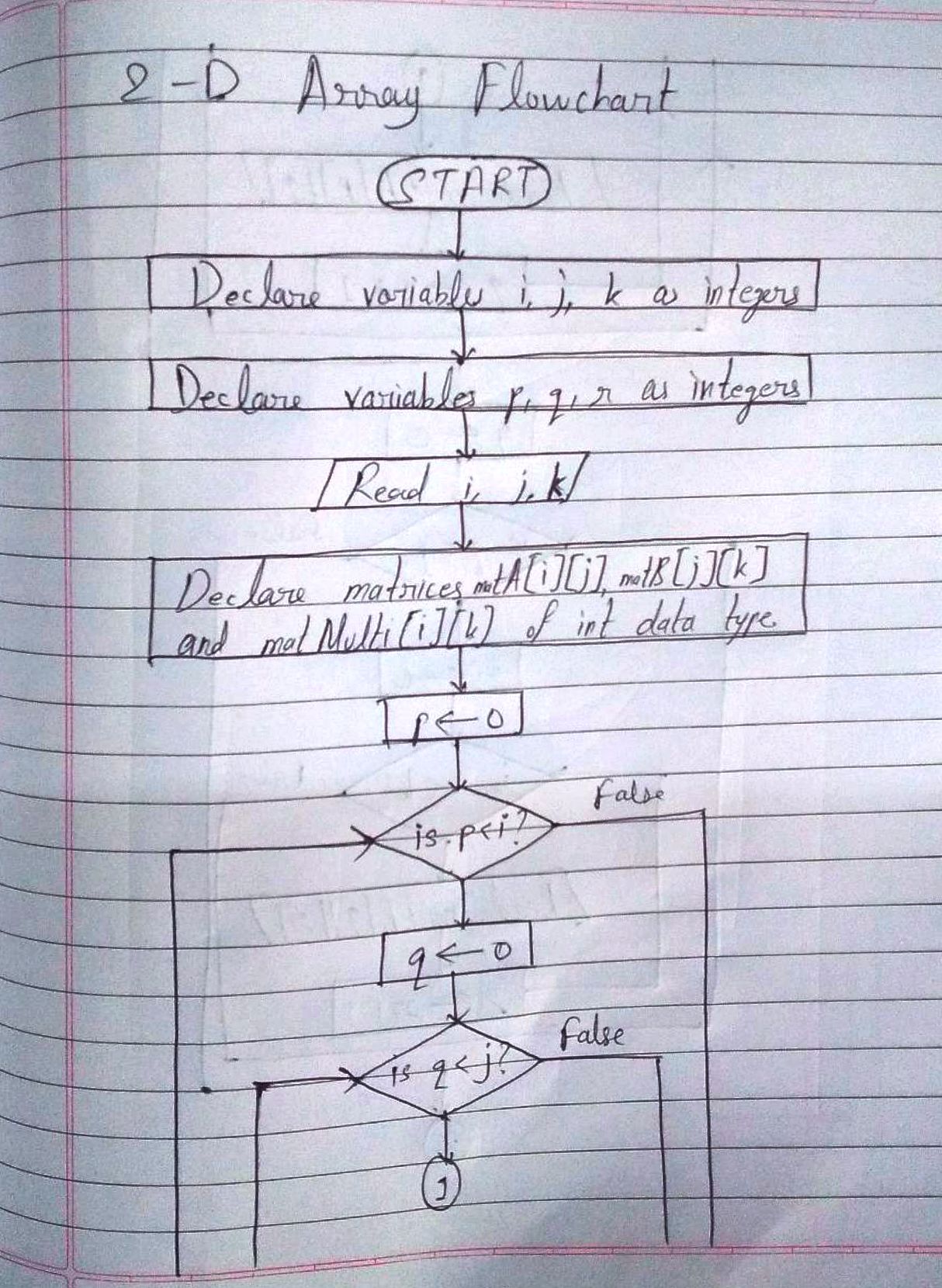
1-D array flowchart:

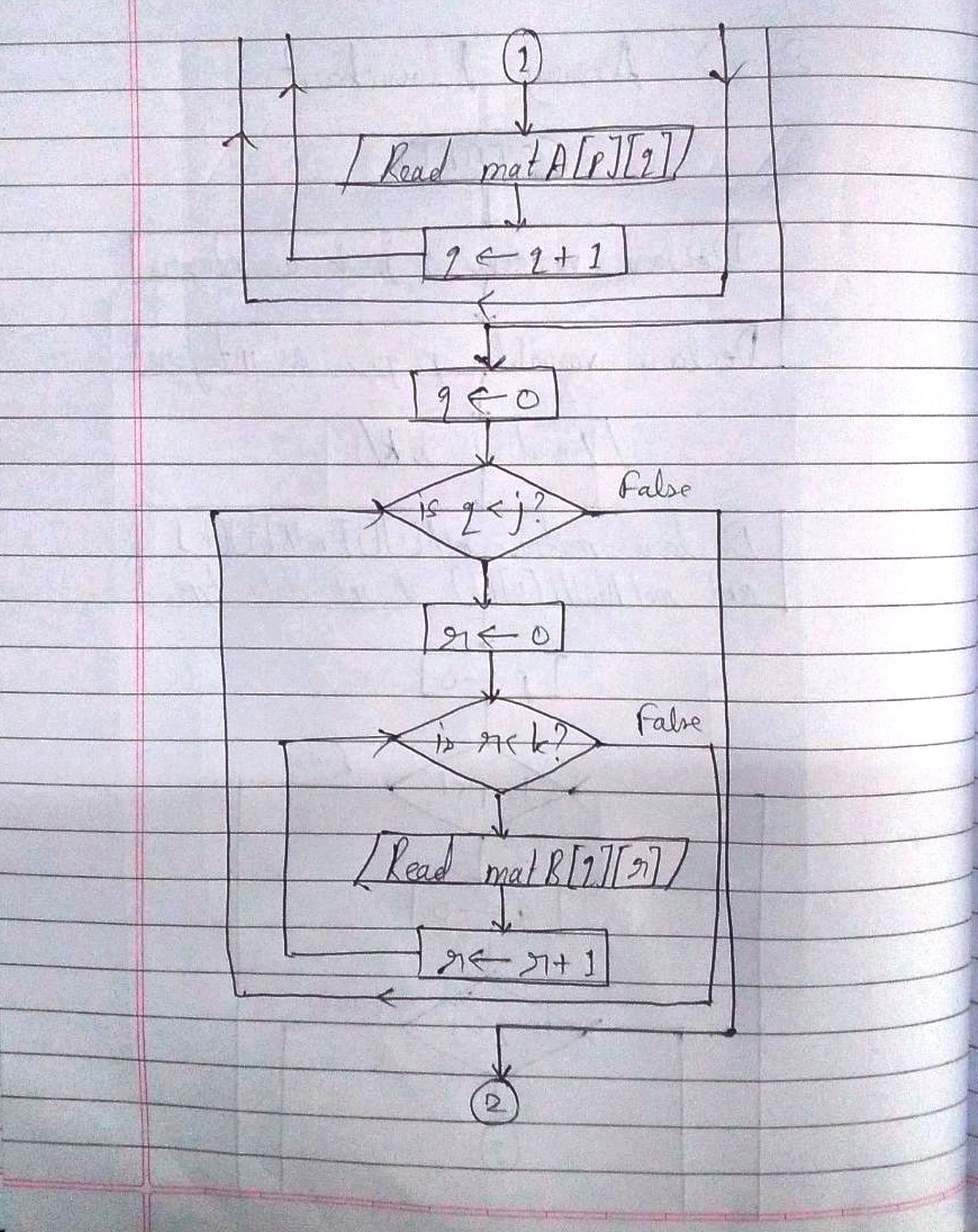


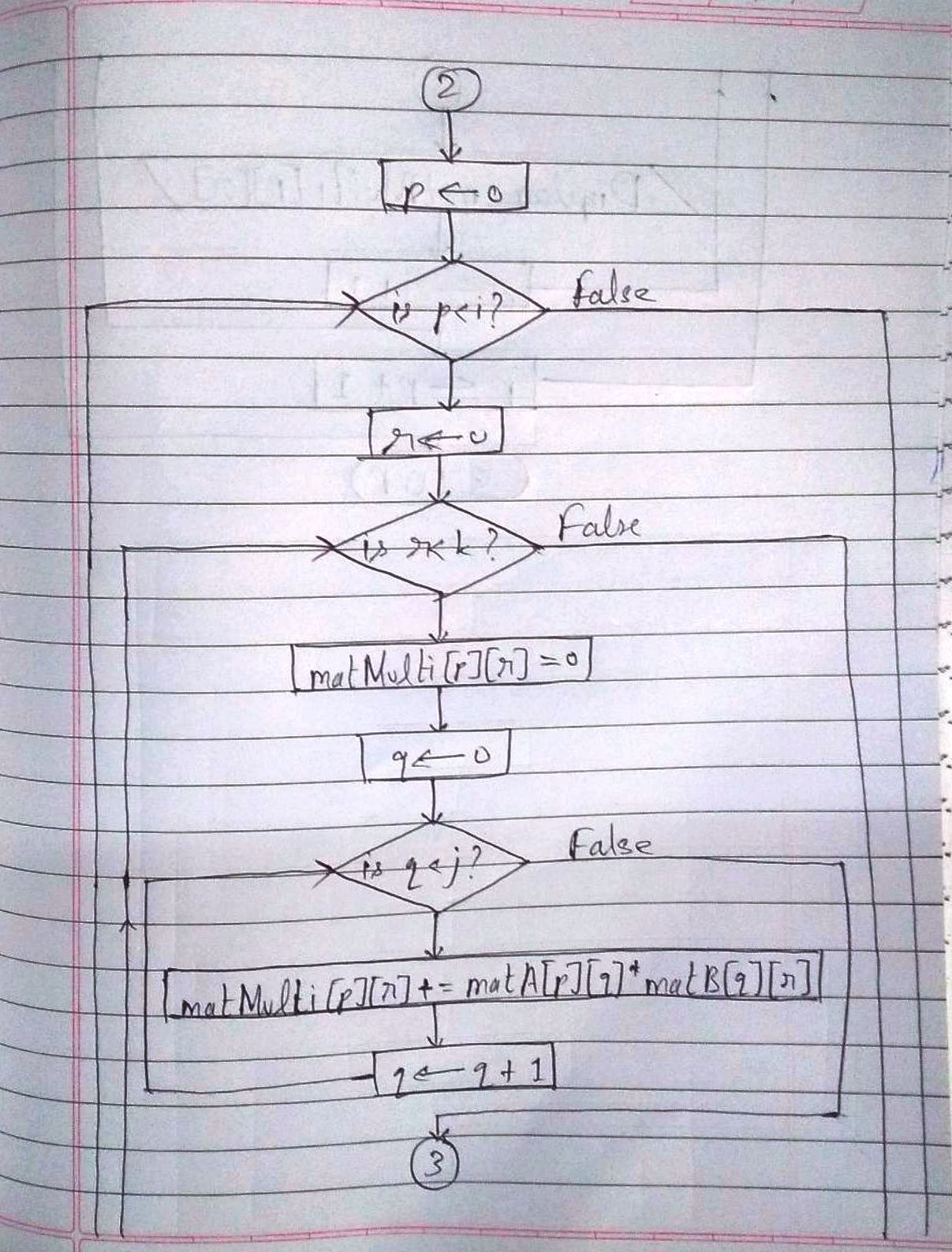


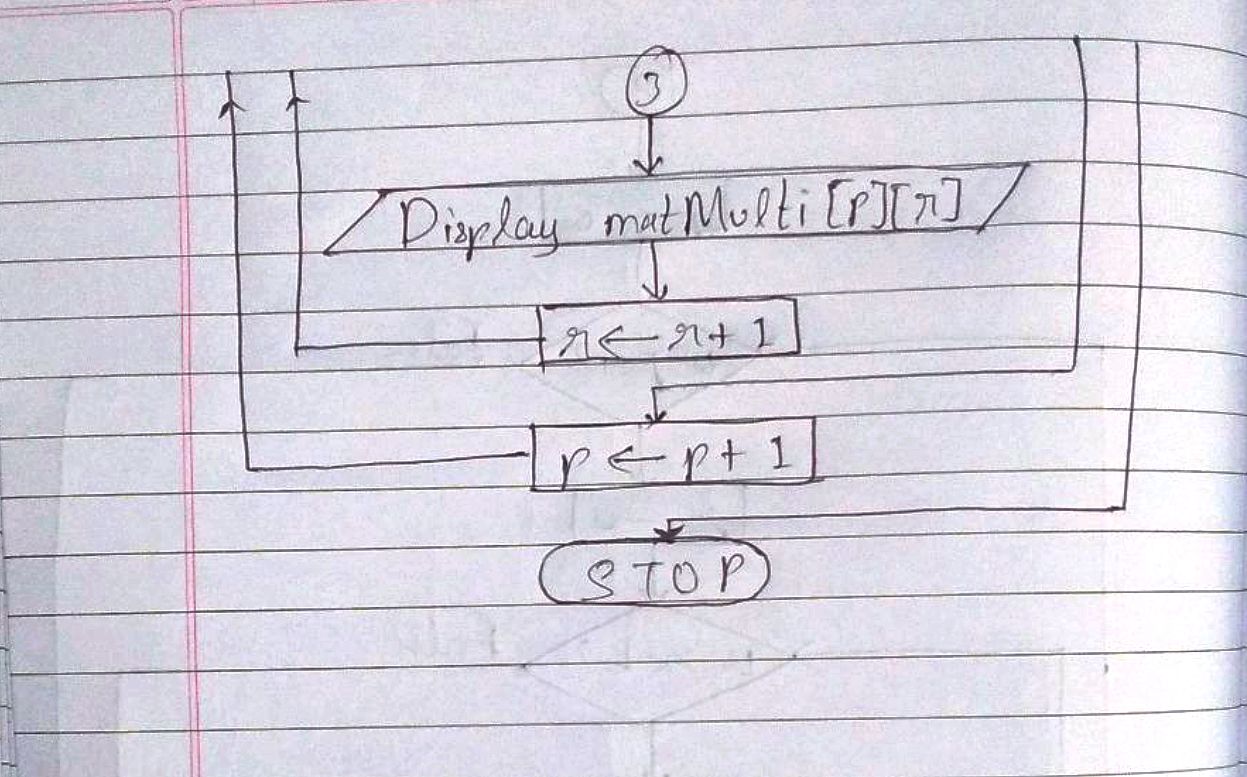


2-D Array Flowchart:









**Implementation details:**

Code as run on Code Blocks:

1-D Array code:

#include<stdio.h>

int main()

{

int i, j, size, key;

printf("\nEnter the size of the array: ");

scanf("%d", &size);

int A[size];

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

{

printf("\nEnter element %d: ", i+1);

scanf("%d", &A[i]);

}

for(i = 1; i < size; i++)

{

key = A[i];

j = i - 1;

while(j>=0 && A[j]>key)

{

A[j+1] = A[j];

j--;

}

A[j+1] = key;

}

printf("The array sorted in ascending order is: ");

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

{

printf("\n%d", A[i]);

}

return 0;

}

2-D Array Code:

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i, j, k;

int p, q, r;

system("cls");

printf("\nEnter the order of matrix A (i by j): ");

scanf("%d", &i);

scanf("%d", &j);

printf("\nEnter the order of matrix B (j by k): ");

scanf("%d", &k);

int matA[i][j];

int matB[j][k];

int matMulti[i][k];

printf("\nEnter the elements of matrix A:");

for(p= 0; p < i; p++)

{

for(q = 0; q < j; q++)

{

printf("\nElement at position [%d][%d] is: ", p, q);

scanf("%d", &matA[p][q]);

}

}

printf("\nEnter the elements of matrix B:");

for(q = 0; q < j; q++)

{

for(r = 0; r < k; r++)

{

printf("\nElement at position [%d][%d] is: ", q, r);

scanf("%d", &matB[q][r]);

}

}

for(p = 0; p < i; p++)

{

for(r = 0; r < k; r++)

{

matMulti[p][r] = 0;

for(q = 0; q < j; q++)

{

matMulti[p][r] += matA[p][q]\*matB[q][r];

}

printf("\n[%d][%d]: %d", p, r, matMulti[p][r]);

}

}

}

Code as copied from Virtual Lab:

1-D Array code:

int main(){

int i, j, size, key ;

int A[size];

for( i = 1 ; i < size ; i++ )

{

key = A[i];

j = i - 1;

while ( j >= 0 && A[j] > key )

{

A[j+1] = A[j];

j--;

}

A[j+1] = key;

}

return 0 ;

}

2-D Array code:

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]

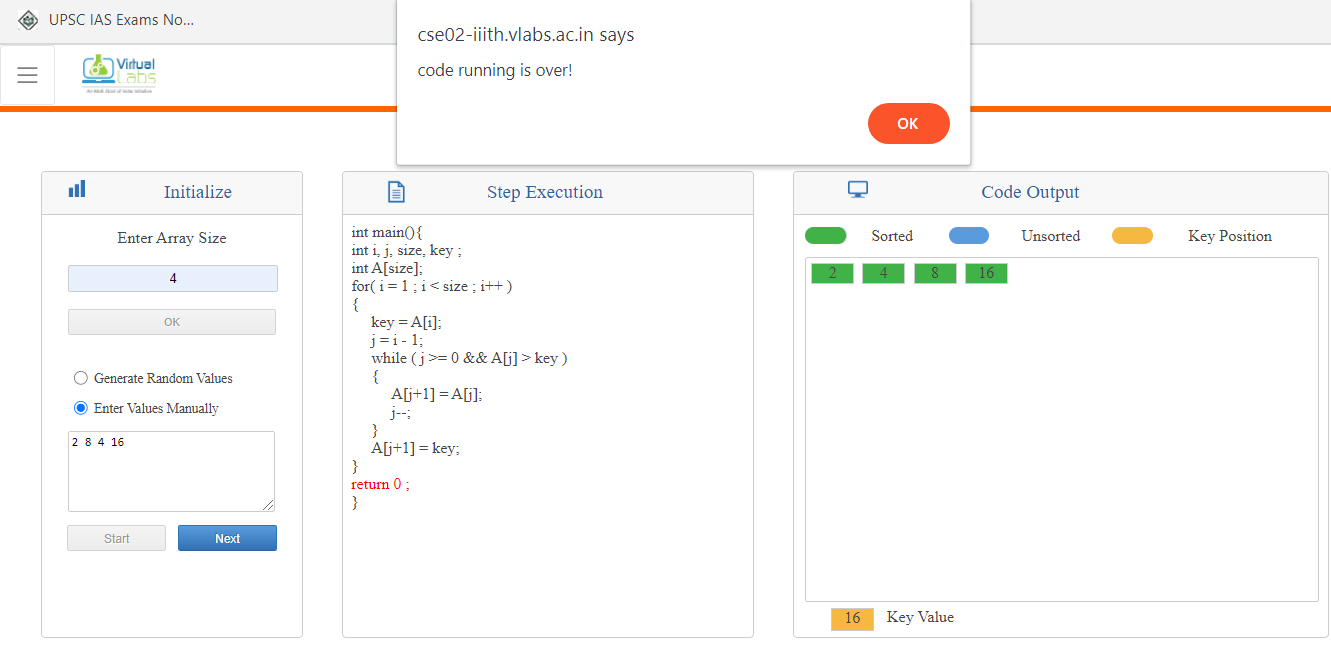
}

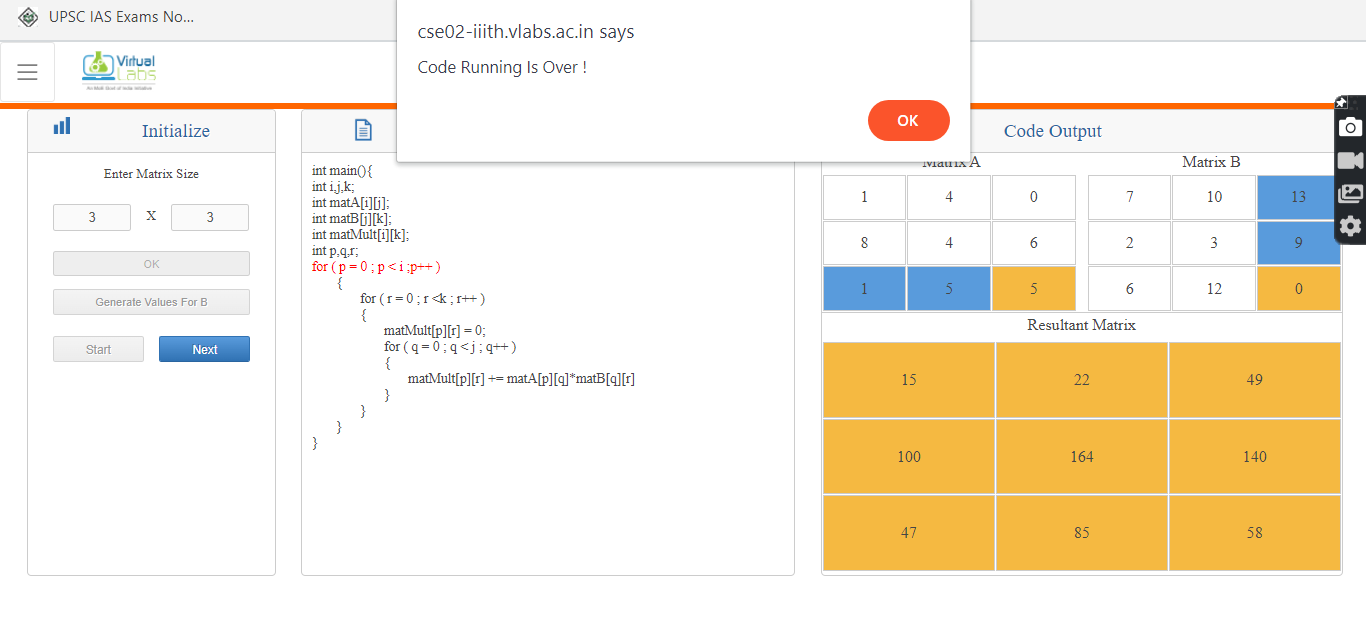
}

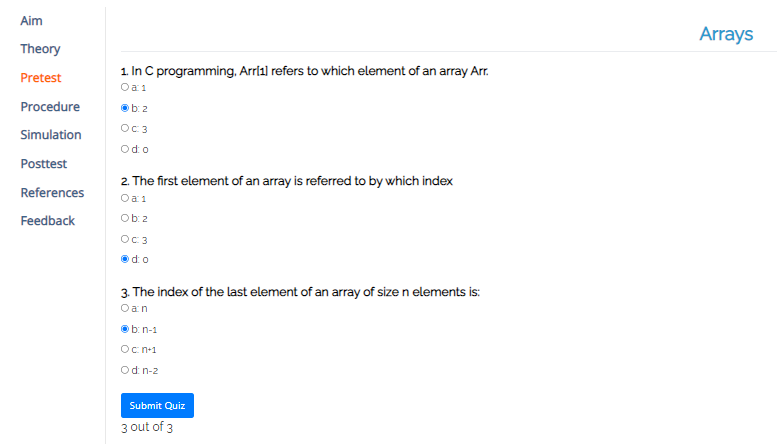
}

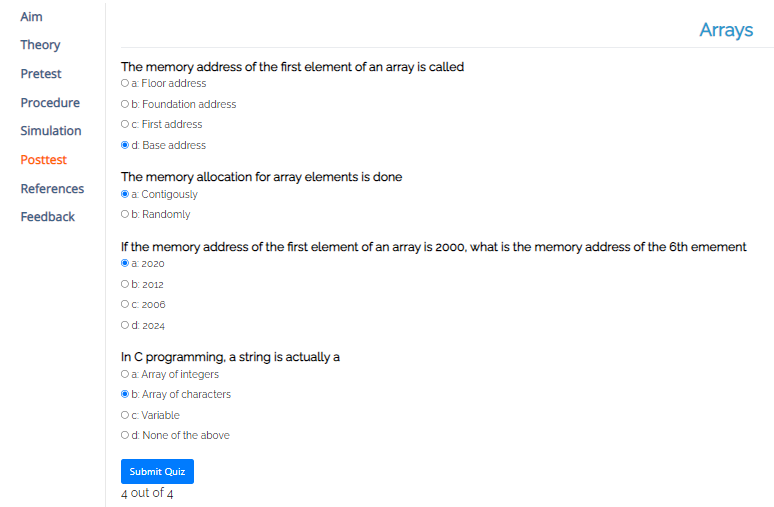
}

**Output(s):**

 Output for 1-D array

 Output for 2-D array

Pretest Quiz

Posttest Quiz

**Conclusion:**

Thus, in this experiment, matrix sorting and matrix multiplication using arrays was performed. By performing this experiment, I understood the logic behind sorting a matrix and multiplying two matrices by a combination of arrays and nested for loops. Thus, it has been observed practically that arrays are better when it comes to storing many values for a single variable, as if some other method was tried instead of using arrays, then it would have been impractical and unscalable.

**Post Lab Descriptive Questions**

1. **Can we change the size of an array at run time? Why or why not?**

**Ans.** The size of an array cannot be changed during the running of a program.

For example, if *i* is a variable then writing code like *int arr[i]* is not valid in C (unless the value of *i* is provided beforehand by the user). Some languages allow the coder to code like that but C does not. This is because if it did, then the stack would become more complicated, function calls would become more expensive and the program would run a lot slower.

1. **Can we pass an array as an argument to a function?**

**Ans.** An array can be passed as an argument to a function. But, only the name of the array is passed, along with square braces, so that the compiler recognizes that an array has been passed as an argument to that function. The array can be of any size, because C does not perform any bound checking on formal parameters.

**Date:** \_\_\_**14-01-2022**\_\_\_\_ **Signature of faculty in-charge**