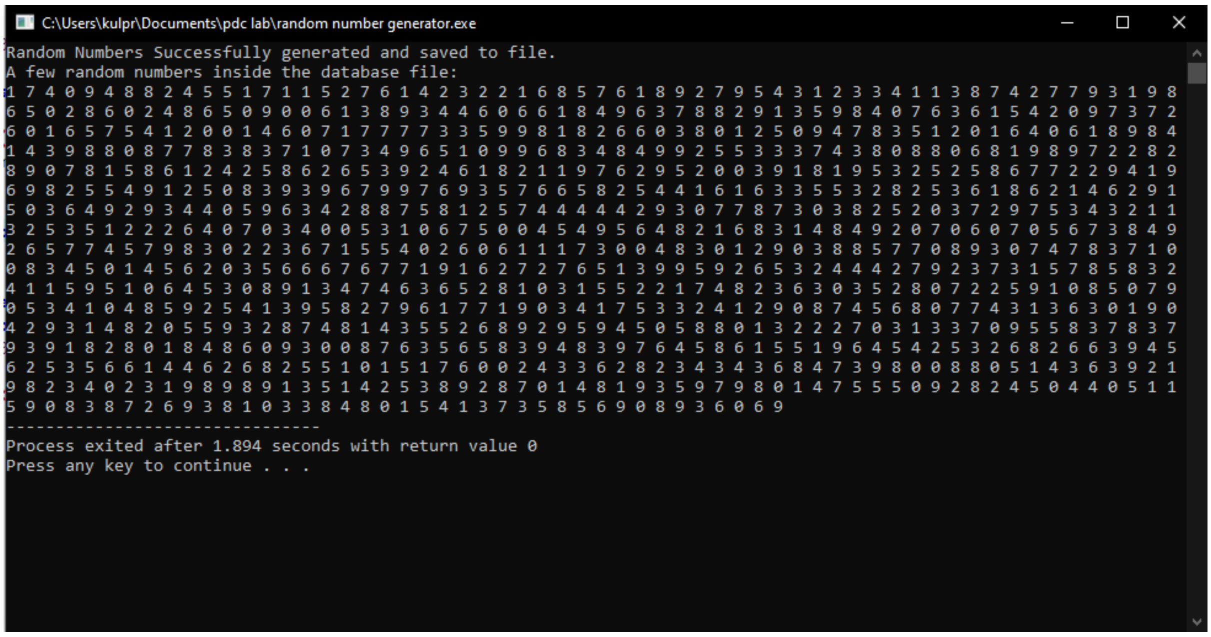
Roll No: 101803186 Name: Kulpreet Singh Group: COE9

UCS545

Lab Assignment I

1. **To generate the random numbers by using different generator parameters and create a file as database to be used for realization of matrix operations.**
   CODE:

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
int main()
{
        int MaxNumber = 10, random;
        ofstream fout;
        fout.open("Database.txt");
        for (int i = 0; i < 16000000; ++i){
                random = rand() % MaxNumber;
                fout << random << " ";
        }
        fout.close();
        cout<<"Random Numbers Successfully generated and saved to file.\n";

        ifstream fin;
        fin.open("Database.txt");
        cout<<"A few random numbers inside the database file:\n";
        for(int i=0;i<1000;i++){
                fin>>random;
                cout<<random<<" ";
        }
        fin.close();
        return 0;
}
```


C:\Users\kulpr\Documents\pdc lab\random number generator.exe

```
Random Numbers Successfully generated and saved to file.
A few random numbers inside the database file:
1 7 4 0 9 4 8 8 2 4 5 5 1 7 1 1 5 2 7 6 1 4 2 3 2 2 1 6 8 5 7 6 1 8 9 2 7 9 5 4 3 1 2 3 3 4 1 1 3 8 7 4 2 7 7 9 3 1 9 8
6 5 0 2 8 6 0 2 4 8 6 5 0 9 0 0 6 1 3 8 9 3 4 4 6 0 6 6 1 8 4 9 6 3 7 8 8 2 9 1 3 5 9 8 4 0 7 6 3 6 1 5 4 2 0 9 7 3 7 2
6 0 1 6 5 7 5 4 1 2 0 0 1 4 6 0 7 1 7 7 7 7 3 3 5 9 9 8 1 8 2 6 6 0 3 8 0 1 2 5 0 9 4 7 8 3 5 1 2 0 1 6 4 0 6 1 8 9 8 4
1 4 3 9 8 8 0 8 7 7 8 3 8 3 7 1 0 7 3 4 9 6 5 1 0 9 9 6 8 3 4 8 4 9 9 2 5 5 3 3 3 7 4 3 8 0 8 8 0 6 8 1 9 8 9 7 2 2 8 2
8 9 0 7 8 1 5 8 6 1 2 4 2 5 8 6 2 6 5 3 9 2 4 6 1 8 2 1 1 9 7 6 2 9 5 2 0 0 3 9 1 8 1 9 5 3 2 5 2 5 8 6 7 7 2 2 9 4 1 9
6 9 8 2 5 5 4 9 1 2 5 0 8 3 9 3 9 6 7 9 9 7 6 9 3 5 7 6 6 5 8 2 5 4 4 1 6 1 6 3 3 5 5 3 2 8 2 5 3 6 1 8 6 2 1 4 6 2 9 1
5 0 3 6 4 9 2 9 3 4 4 0 5 9 6 3 4 2 8 8 7 5 8 1 2 5 7 4 4 4 4 4 2 9 3 0 7 7 8 7 3 0 3 8 2 5 2 0 3 7 2 9 7 5 3 4 3 2 1 1
3 2 5 3 5 1 2 2 2 6 4 0 7 0 3 4 0 0 5 3 1 0 6 7 5 0 0 4 5 4 9 5 6 4 8 2 1 6 8 3 1 4 8 4 9 2 0 7 0 6 0 7 0 5 6 7 3 8 4 9
2 6 5 7 7 4 5 7 9 8 3 0 2 2 3 6 7 1 5 5 4 0 2 6 0 6 1 1 1 7 3 0 0 4 8 3 0 1 2 9 0 3 8 8 5 7 7 0 8 9 3 0 7 4 7 8 3 7 1 0
0 8 3 4 5 0 1 4 5 6 2 0 3 5 6 6 6 7 6 7 7 1 9 1 6 2 7 2 7 6 5 1 3 9 9 5 9 2 6 5 3 2 4 4 4 2 7 9 2 3 7 3 1 5 7 8 5 8 3 2
4 1 1 5 9 5 1 0 6 4 5 3 0 8 9 1 3 4 7 4 6 3 6 5 2 8 1 0 3 1 5 5 2 2 1 7 4 8 2 3 6 3 0 3 5 2 8 0 7 2 2 5 9 1 0 8 5 0 7 9
0 5 3 4 1 0 4 8 5 9 2 5 4 1 3 9 5 8 2 7 9 6 1 7 7 1 9 0 3 4 1 7 5 3 3 2 4 1 2 9 0 8 7 4 5 6 8 0 7 7 4 3 1 3 6 3 0 1 9 0
4 2 9 3 1 4 8 2 0 5 5 9 3 2 8 7 4 8 1 4 3 5 5 2 6 8 9 2 9 5 9 4 5 0 5 8 8 0 1 3 2 2 2 7 0 3 1 3 3 7 0 9 5 5 8 3 7 8 3 7
9 3 9 1 8 2 8 0 1 8 4 8 6 0 9 3 0 0 8 7 6 3 5 6 5 8 3 9 4 8 3 9 7 6 4 5 8 6 1 5 5 1 9 6 4 5 4 2 5 3 2 6 8 2 6 6 3 9 4 5
6 2 5 3 5 6 6 1 4 4 6 2 6 8 2 5 5 1 0 1 5 1 7 6 0 0 2 4 3 3 6 2 8 2 3 4 3 4 3 6 8 4 7 3 9 8 0 0 8 8 0 5 1 4 3 6 3 9 2 1
9 8 2 3 4 0 2 3 1 9 8 9 8 9 1 3 5 1 4 2 5 3 8 9 2 8 7 0 1 4 8 1 9 3 5 9 7 9 8 0 1 4 7 5 5 5 0 9 2 8 2 4 5 0 4 4 0 5 1 1
5 9 0 8 3 8 7 2 6 9 3 8 1 0 3 3 8 4 8 0 1 5 4 1 3 7 3 5 8 5 6 9 0 8 9 3 6 0 6 9
--------------------------------
Process exited after 1.894 seconds with return value 0
Press any key to continue . . .
```

2. **Perform the matrix basic operations (Addition, Subtraction Multiply, Matrix Inversion, (a matrix and a vector)), for different dimensions of the matrix.**
   CODE:

```
#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

void Cofactor(int **A, int **temp, int p, int q, int n)
{
	int i = 0, j = 0;

	for (int row = 0; row < n; row++)
	{
		for (int col = 0; col < n; col++)
		{
			if (row != p && col != q)
			{
				temp[i][j++] = A[row][col];

				if (j == n - 1)
				{
					j = 0;
					i++;
				}
			}
		}
	}
}

int Determinant(int **A, int n)
{
	int D = 0;
	if (n == 1)
		return A[0][0];

	int **temp = new int*[100];
	for (int i = 0; i < 100; ++i)
	{
		temp[i] = new int[100];
	}

	int sign = 1;

	for (int f = 0; f < n; f++)
	{
		Cofactor(A, temp, 0, f, n);
```

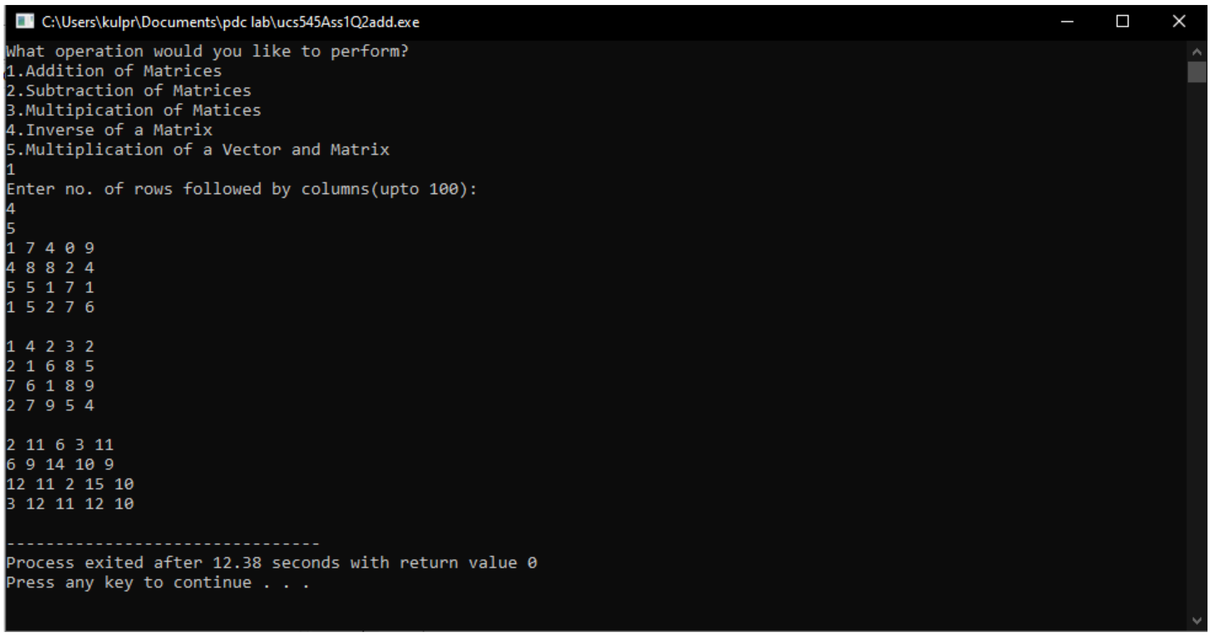
```
			D += sign * A[0][f] * Determinant(temp, n - 1);

			sign = -sign;
		}

		return D;
	}

	void adjoint(int **A, int **adj, int N)
	{
		if (N == 1)
		{
			adj[0][0] = 1;
			return;
		}
		int sign = 1;
		int** temp = new int* [100];
		for (int i = 0; i < 100; ++i)
		{
			temp[i] = new int[100];
		}


		for (int i = 0; i < N; i++)
		{
			for (int j = 0; j < N; j++)
			{
				Cofactor(A, temp, i, j, N);

				sign = ((i + j) % 2 == 0) ? 1 : -1;

				adj[j][i] = (sign) * (Determinant(temp, N - 1));
			}
		}
	}

	int main()
	{
		ifstream temp;
		temp.open("Database.txt");

		const int m = 100;

		int** matrixA = new int* [m];
		int** matrixB = new int* [m];
		int** Answer = new int* [m];
		float** InverseMat = new float* [m];
	for (int i = 0; i < m; ++i)
	{
		matrixA[i] = new int[100];
		matrixB[i] = new int[100];
		Answer[i] = new int[100];
		InverseMat[i] = new float[100];
	}

	for (int i = 0; i < m; ++i)
	{
		for (int j = 0; j < m; ++j)
		{
			matrixA[i][j] = 0;
			matrixB[i][j] = 0;
			Answer[i][j] = 0;
		}
	}
	int a, r1 = 0, c1 = 0, r2 = 0, c2 = 0, determinant=0;


	cout << "What operation would you like to perform?\n1.Addition of
Matrices\n2.Subtraction of Matrices\n3.Multipication of Matices\n4.Inverse of a
Matrix\n5.Multiplication of a Vector and Matrix" << endl;
	cin >> a;

	switch (a)
	{
		case 1: cout << "Enter no. of rows followed by columns(upto 100):" << endl;
			cin >> r1 >> c1;
			r2 = r1;
			c2 = c1;

			for (int i = 0; i < r1; ++i)
			{
				for (int j = 0; j < c1; ++j)
				{
					temp >> matrixA[i][j];
					cout << matrixA[i][j] << " ";
				}
				cout << "\n";
			}
			cout << "\n";
			for (int i = 0; i < r2; ++i)
			{
				for (int j = 0; j < c2; ++j)
				{
					temp >> matrixB[i][j];
					cout << matrixB[i][j] << " ";
```

```
                        }
                        cout << "\n";
                }
                cout << "\n";
                for (int i = 0; i < r1; ++i)
                {
                        for (int j = 0; j < c1; ++j)
                        {
                                Answer[i][j] = matrixA[i][j]+matrixB[i][j];
                                cout << Answer[i][j] << " ";
                        }
                        cout << "\n";
                }
                break;

        case 2: cout << "Enter no. of rows followed by columns(upto 100):" << endl;
                cin >> r1 >> c1;
                r2 = r1;
                c2 = c1;

                for (int i = 0; i < r1; ++i)
                {
                        for (int j = 0; j < c1; ++j)
                        {
                                temp >> matrixA[i][j];
                                cout << matrixA[i][j] << " ";
                        }
                        cout << "\n";
                }
                cout << "\n";
                for (int i = 0; i < r2; ++i)
                {
                        for (int j = 0; j < c2; ++j)
                        {
                                temp >> matrixB[i][j];
                                cout << matrixB[i][j] << " ";
                        }
                        cout << "\n";
                }
                cout << "\n";
                for (int i = 0; i < r1; ++i)
                {
                        for (int j = 0; j < c1; ++j)
                        {
                                Answer[i][j] = matrixA[i][j] - matrixB[i][j];
                                cout << Answer[i][j] << " ";
                        }
                        cout << "\n";
```

```
                    }
                    break;

                case 3: cout << "Enter no. of rows(Matrix A)(upto 100) followed by
        columns(Matrix A)(upto 100) followed by columns(Matrix B)(upto 100):" << endl;
                    cin >> r1 >> c1 >> c2;
                    r2 = c1;

                    for (int i = 0; i < r1; ++i)
                    {
                        for (int j = 0; j < c1; ++j)
                        {
                            temp >> matrixA[i][j];
                            cout << matrixA[i][j] << " ";
                        }
                        cout << "\n";
                    }
                    cout << "\n";
                    for (int i = 0; i < r2; ++i)
                    {
                        for (int j = 0; j < c2; ++j)
                        {
                            temp >> matrixB[i][j];
                            cout << matrixB[i][j] << " ";
                        }
                        cout << "\n";
                    }
                    cout << "\n";

                    for (int i = 0; i < r1; ++i)
                    {
                        for (int j = 0; j < c2; ++j)
                        {
                            for (int k = 0; k < r2; ++k)
                            {
                                Answer[i][j] += matrixA[i][k] * matrixB[k][j];
                            }
                            cout << Answer[i][j] << " ";
                        }
                        cout << "\n";
                    }
                    break;

                case 4: cout << "Enter the Row = Column size of square matrix:" << endl;
                    cin >> r1;
                    c1 = r1;

                    for (int i = 0; i < r1; ++i)
                {
                        for (int j = 0; j < c1; ++j)
                        {
                                temp >> matrixA[i][j];
                                cout << matrixA[i][j] << " ";
                        }
                        cout << "\n";
                }
                cout << "\n";

                determinant = Determinant(matrixA, r1);
                cout <<"Determinant is: " << determinant << endl;

                if (determinant == 0)
                {
                        cout << "Inverse does not exist!" << endl;
                        break;
                }

                adjoint(matrixA, matrixB, r1);
                cout<<"Inverse Matrix is: "<<endl;
                for (int i = 0; i < r1; ++i)
                {
                        for (int j = 0; j < r1; ++j)
                        {
                                InverseMat[i][j] = matrixB[i][j] / (float)determinant;
                                cout << InverseMat[i][j] << " ";
                        }
                        cout << "\n";
                }
                break;

        case 5: cout << "Vector into Matrix(enter 1) or Matrix into Vector(enter 2)?"
<< endl;
                int b;
                cin >> b;
                if (b == 1)
                {
                        cout << "Enter dimensions of the matrix(rows followed by
columns):" << endl;
                        cin >> r2 >> c2;
                        r1 = 1;
                        c1 = r2;
                }
                else if (b == 2)
                {
                        cout << "Enter dimensions of the matrix(rows followed by
columns):" << endl;
```

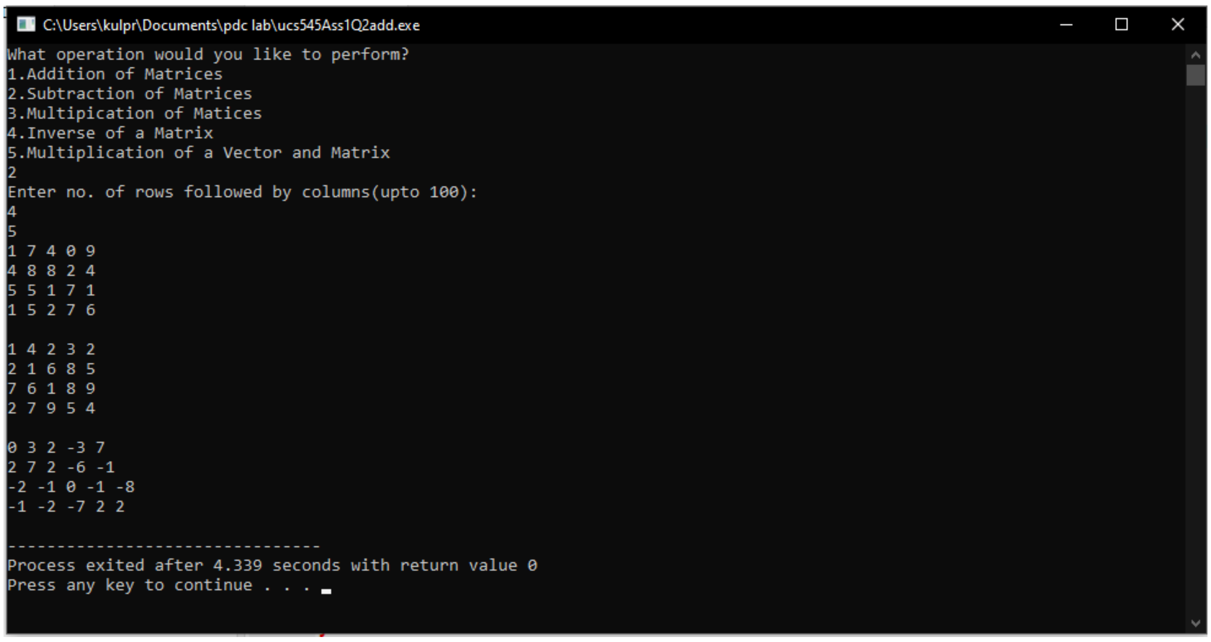
```
                cin >> r1 >> c1;
                r2 = c1;
                c2 = 1;
            }
            else {
                cout << "Invalid input!" << endl;
                break;
            }

            for (int i = 0; i < r1; ++i)
            {
                for (int j = 0; j < c1; ++j)
                {
                    temp >> matrixA[i][j];
                    cout << matrixA[i][j] << " ";
                }
                cout << "\n";
            }
            cout << "\n";
            for (int i = 0; i < r2; ++i)
            {
                for (int j = 0; j < c2; ++j)
                {
                    temp >> matrixB[i][j];
                    cout << matrixB[i][j] << " ";
                }
                cout << "\n";
            }
            cout << "\n";

            for (int i = 0; i < r1; ++i)
            {
                for (int j = 0; j < c2; ++j)
                {
                    for (int k = 0; k < r2; ++k)
                    {
                        Answer[i][j] += matrixA[i][k] * matrixB[k][j];
                    }
                    cout << Answer[i][j] << " ";
                }
                cout << "\n";
            }
            break;

        default: cout << "Invalid Input!" << endl;
            break;
    }
```

```
            temp.close();

            return 0;
}
```



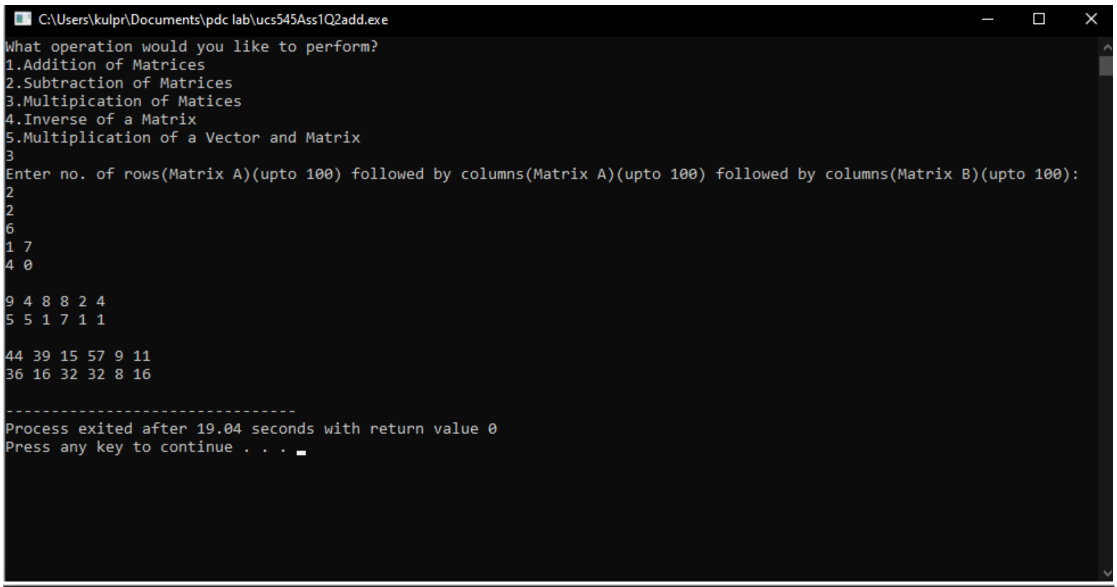
C:\Users\kulpr\Documents\pdc lab\ucs545Ass1Q2add.exe

```
What operation would you like to perform?
1.Addition of Matrices
2.Subtraction of Matrices
3.Multipication of Matices
4.Inverse of a Matrix
5.Multiplication of a Vector and Matrix
1
Enter no. of rows followed by columns(upto 100):
4
5
1 7 4 0 9
4 8 8 2 4
5 5 1 7 1
1 5 2 7 6

1 4 2 3 2
2 1 6 8 5
7 6 1 8 9
2 7 9 5 4

2 11 6 3 11
6 9 14 10 9
12 11 2 15 10
3 12 11 12 10

--------------------------------
Process exited after 12.38 seconds with return value 0
Press any key to continue . . .
```

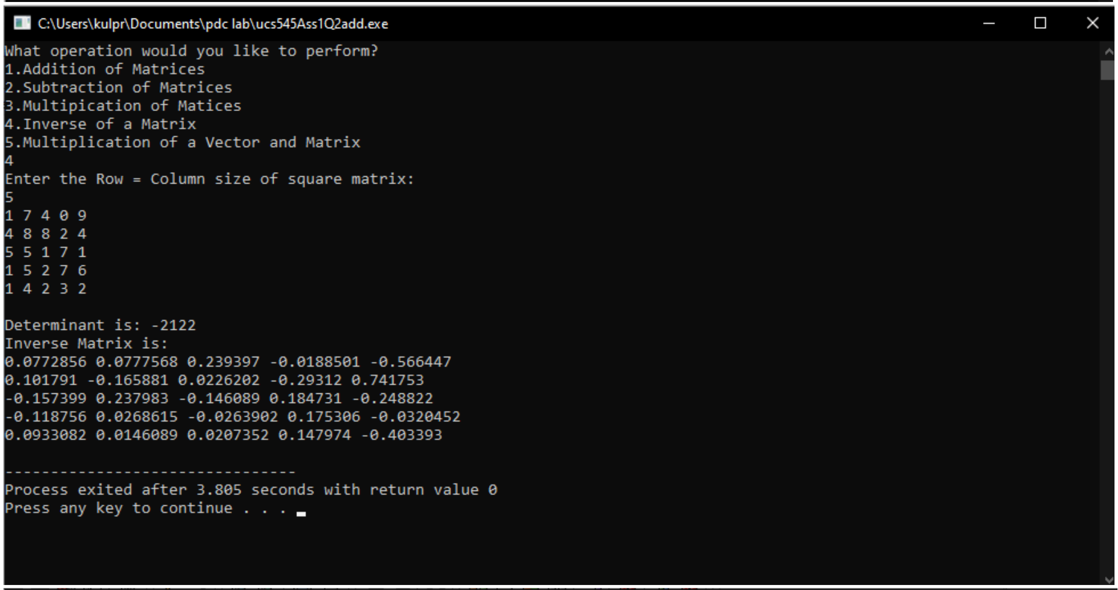


C:\Users\kulpr\Documents\pdc lab\ucs545Ass1Q2add.exe

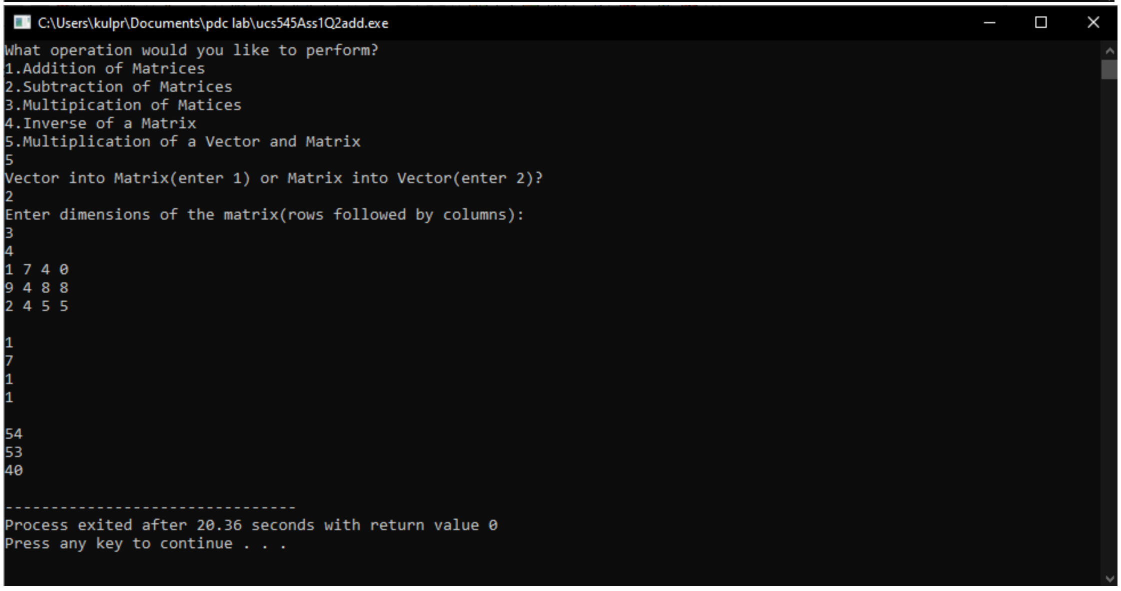
```
What operation would you like to perform?
1.Addition of Matrices
2.Subtraction of Matrices
3.Multipication of Matices
4.Inverse of a Matrix
5.Multiplication of a Vector and Matrix
2
Enter no. of rows followed by columns(upto 100):
4
5
1 7 4 0 9
4 8 8 2 4
5 5 1 7 1
1 5 2 7 6

1 4 2 3 2
2 1 6 8 5
7 6 1 8 9
2 7 9 5 4

0 3 2 -3 7
2 7 2 -6 -1
-2 -1 0 -1 -8
-1 -2 -7 2 2

--------------------------------
Process exited after 4.339 seconds with return value 0
Press any key to continue . . .
```



```
What operation would you like to perform?
1.Addition of Matrices
2.Subtraction of Matrices
3.Multipication of Matices
4.Inverse of a Matrix
5.Multiplication of a Vector and Matrix
3
Enter no. of rows(Matrix A)(upto 100) followed by columns(Matrix A)(upto 100) followed by columns(Matrix B)(upto 100):
2
2
6
1 7
4 0

9 4 8 8 2 4
5 5 1 7 1 1

44 39 15 57 9 11
36 16 32 32 8 16

--------------------------------
Process exited after 19.04 seconds with return value 0
Press any key to continue . . .
```



```
What operation would you like to perform?
1.Addition of Matrices
2.Subtraction of Matrices
3.Multipication of Matices
4.Inverse of a Matrix
5.Multiplication of a Vector and Matrix
4
Enter the Row = Column size of square matrix:
5
1 7 4 0 9
4 8 8 2 4
5 5 1 7 1
1 5 2 7 6
1 4 2 3 2

Determinant is: -2122
Inverse Matrix is:
0.0772856 0.0777568 0.239397 -0.0188501 -0.566447
0.101791 -0.165881 0.0226202 -0.29312 0.741753
-0.157399 0.237983 -0.146089 0.184731 -0.248822
-0.118756 0.0268615 -0.0263902 0.175306 -0.0320452
0.0933082 0.0146089 0.0207352 0.147974 -0.403393

--------------------------------
Process exited after 3.805 seconds with return value 0
Press any key to continue . . .
```

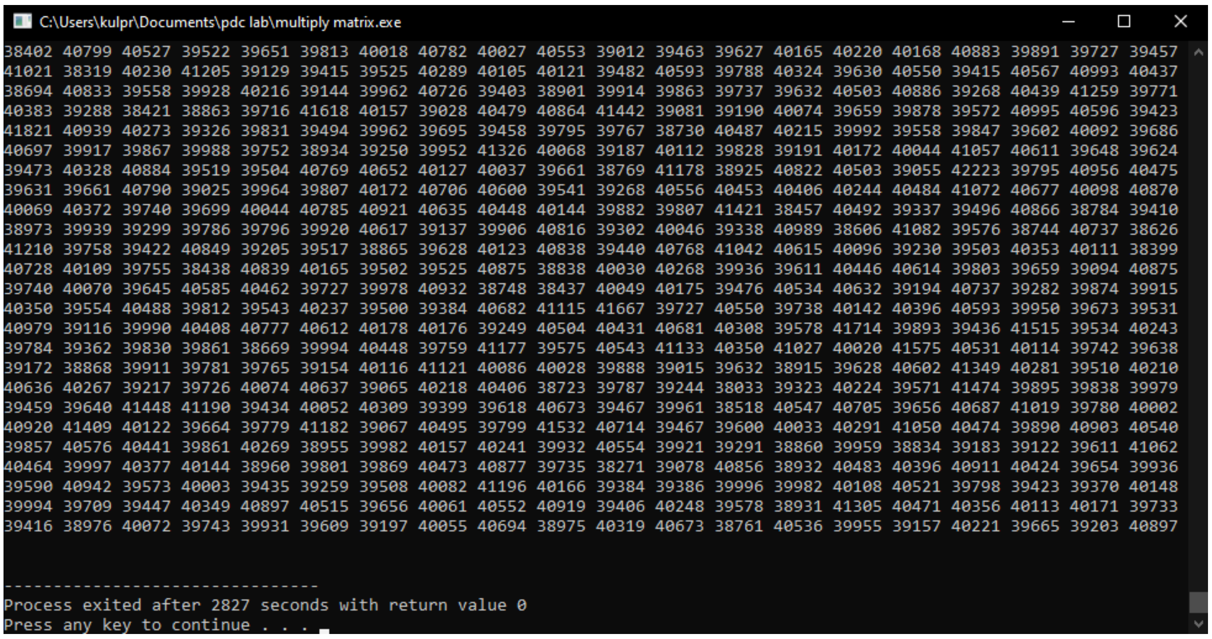


```
What operation would you like to perform?
1.Addition of Matrices
2.Subtraction of Matrices
3.Multipication of Matices
4.Inverse of a Matrix
5.Multiplication of a Vector and Matrix
5
Vector into Matrix(enter 1) or Matrix into Vector(enter 2)?
2
Enter dimensions of the matrix(rows followed by columns):
3
4
1 7 4 0
9 4 8 8
2 4 5 5

1
7
1
1

54
53
40

--------------------------------
Process exited after 20.36 seconds with return value 0
Press any key to continue . . .
```

3. **Multiply 2 matrices A[2000,2000] and B[2000,6000]. Calculate the computational time.**

CODE:

```
#include<bits/stdc++.h>
#include<fstream>
using namespace std;

int main(){
        ifstream temp;
        temp.open("Database.txt");

        const int m = 2000;
        const int n = 2000;
        const int p = 6000;

        int** a = new int*[m];
        for(int i=0;i<m;i++){
                a[i] = new int[n];
                for(int j=0;j<n;j++){
                        temp >> a[i][j];
                }
        }
        int** b = new int*[n];
        for(int i=0;i<n;i++){
                b[i] = new int[p];
                for(int j=0;j<p;j++){
                        temp >> b[i][j];
                }
        }

        int** c = new int*[m];
        for(int i=0;i<m;i++){
                c[i] = new int[p];
                for(int j=0;j<p;j++){
                        c[i][j] = 0;
                }
        }

        for(int i=0;i<m;i++){
                for(int j=0;j<p;j++){
                        for(int k=0;k<n;k++){
                                c[i][j] += a[i][k] * b[k][j];
                        }
                        cout<<c[i][j]<<" ";
                }
                cout<<endl;
        }
        temp.close();

        return 0;
}
```



C:\Users\kulpr\Documents\pdc lab\multiply matrix.exe

```
38402 40799 40527 39522 39651 39813 40018 40782 40027 40553 39012 39463 39627 40165 40220 40168 40883 39891 39727 39457
41021 38319 40230 41205 39129 39415 39525 40289 40105 40121 39482 40593 39788 40324 39630 40550 39415 40567 40993 40437
38694 40833 39558 39928 40216 39144 39962 40726 39403 38901 39914 39863 39737 39632 40503 40886 39268 40439 41259 39771
40383 39288 38421 38863 39716 41618 40157 39028 40479 40864 41442 39081 39190 40074 39659 39878 39572 40995 40596 39423
41821 40939 40273 39326 39831 39494 39962 39695 39458 39795 39767 38730 40487 40215 39992 39558 39847 39602 40092 39686
40697 39917 39867 39988 39752 38934 39250 39952 41326 40068 39187 40112 39828 39191 40172 40044 41057 40611 39648 39624
39473 40328 40884 39519 39504 40769 40652 40127 40037 39661 38769 41178 38925 40822 40503 39055 42223 39795 40956 40475
39631 39661 40790 39025 39964 39807 40172 40706 40600 39541 39268 40556 40453 40406 40244 40484 41072 40677 40098 40870
40069 40372 39740 39699 40044 40785 40921 40635 40448 40144 39882 39807 41421 38457 40492 39337 39496 40866 38784 39410
38973 39939 39299 39786 39796 39920 40617 39137 39906 40816 39302 40046 39338 40989 38606 41082 39576 38744 40737 38626
41210 39758 39422 40849 39205 39517 38865 39628 40123 40838 39440 40768 41042 40615 40096 39230 39503 40353 40111 38399
40728 40109 39755 38438 40839 40165 39502 39525 40875 38838 40030 40268 39936 39611 40446 40614 39803 39659 39094 40875
39740 40070 39645 40585 40462 39727 39978 40932 38748 38437 40049 40175 39476 40534 40632 39194 40737 39282 39874 39915
40350 39554 40488 39812 39543 40237 39500 39384 40682 41115 41667 39727 40550 39738 40142 40396 40593 39950 39673 39531
40979 39116 39990 40408 40777 40612 40178 40176 39249 40504 40431 40681 40308 39578 41714 39893 39436 41515 39534 40243
39784 39362 39830 39861 38669 39994 40448 39759 41177 39575 40543 41133 40350 41027 40020 41575 40531 40114 39742 39638
39172 38868 39911 39781 39765 39154 40116 41121 40086 40028 39888 39015 39632 38915 39628 40602 41349 40281 39510 40210
40636 40267 39217 39726 40074 40637 39065 40218 40406 38723 39787 39244 38033 39323 40224 39571 41474 39895 39838 39979
39459 39640 41448 41190 39434 40052 40309 39399 39618 40673 39467 39961 38518 40547 40705 39656 40687 41019 39780 40002
40920 41409 40122 39664 39779 41182 39067 40495 39799 41532 40714 39467 39600 40033 40291 41050 40474 39890 40903 40540
39857 40576 40441 39861 40269 38955 39982 40157 40241 39932 40554 39921 39291 38860 39959 38834 39183 39122 39611 41062
40464 39997 40377 40144 38960 39801 39869 40473 40877 39735 38271 39078 40856 38932 40483 40396 40911 40424 39654 39936
39590 40942 39573 40003 39435 39259 39508 40082 41196 40166 39384 39386 39996 39982 40108 40521 39798 39423 39370 40148
39994 39709 39447 40349 40897 40515 39656 40061 40552 40919 39406 40248 39578 38931 41305 40471 40356 40113 40171 39733
39416 38976 40072 39743 39931 39609 39197 40055 40694 38975 40319 40673 38761 40536 39955 39157 40221 39665 39203 40897


--------------------------------
Process exited after 2827 seconds with return value 0
Press any key to continue . . .
```

Therefore, the **total process execution time/computational time** is **2827 seconds.**

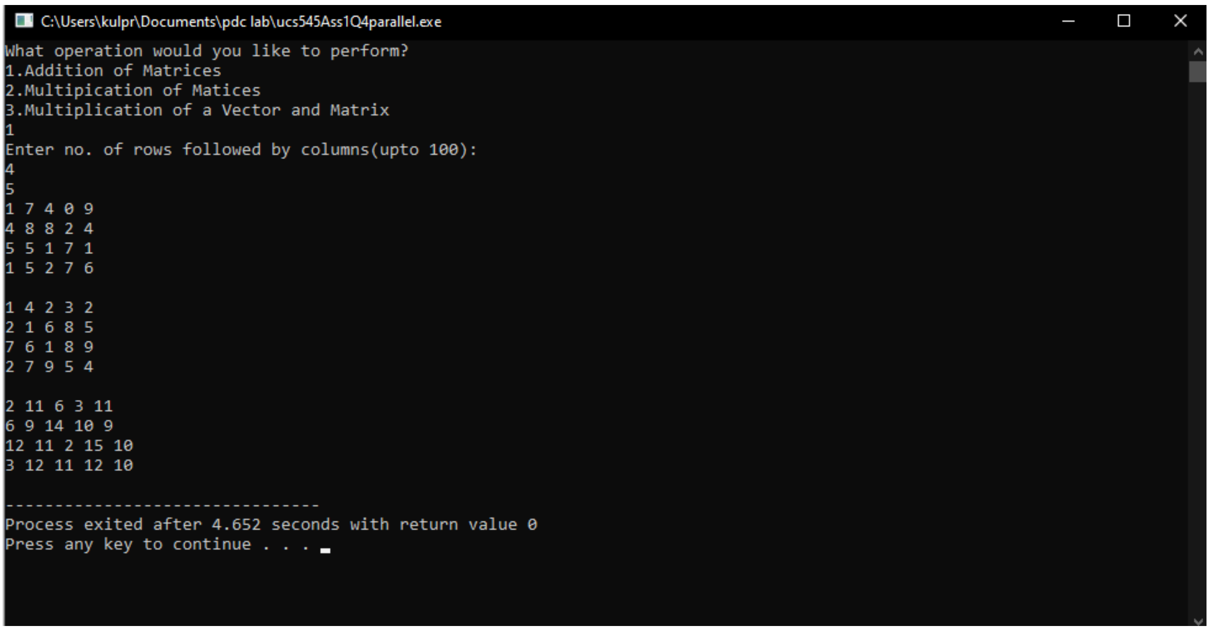
4. **Perform parallel program for the matrix operations (Addition, Multiply, (matrix and vector)). Calculate the computational time.**

CODE:

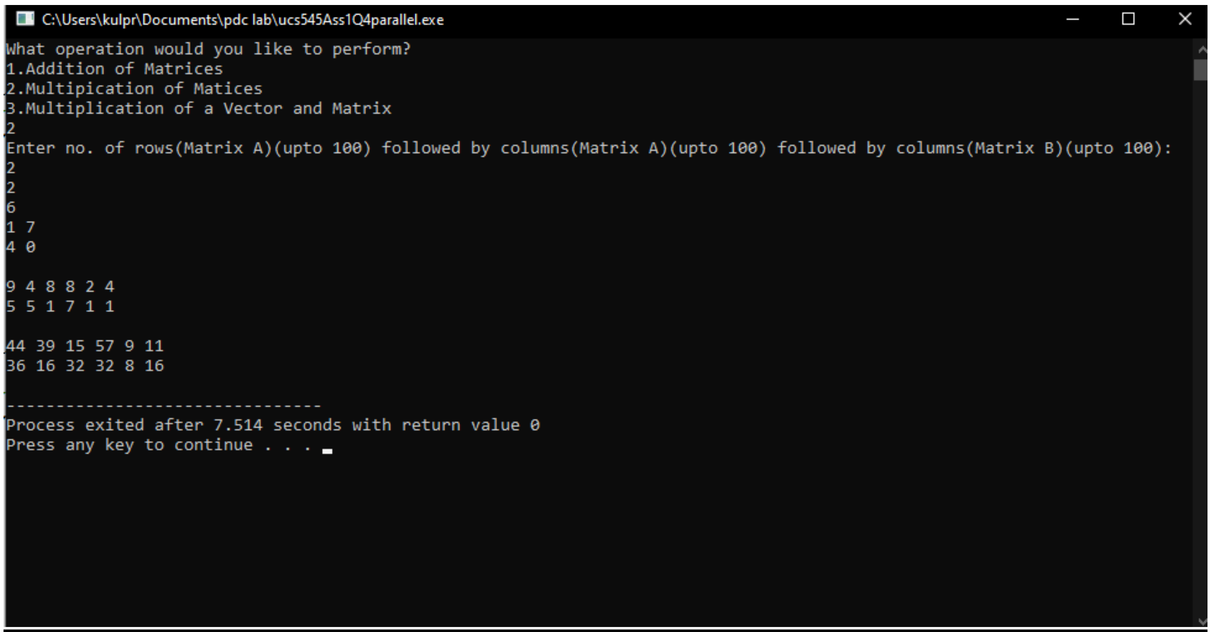
```
#include <iostream>
#include <fstream>
#include <cmath>
#include<omp.h>
using namespace std;

int main()
{
        ifstream temp;
        temp.open("Database.txt");

        const int m = 100;

        int** matrixA = new int* [m];
        int** matrixB = new int* [m];
        int** Answer = new int* [m];

        for (int i = 0; i < m; ++i)
        {
                matrixA[i] = new int[100];
                matrixB[i] = new int[100];
```

```
		Answer[i] = new int[100];
	}
	#pragma omp parallel for
	for (int i = 0; i < m; ++i)
	{
		for (int j = 0; j < m; ++j)
		{
			matrixA[i][j] = 0;
			matrixB[i][j] = 0;
			Answer[i][j] = 0;
		}
	}
	int a, r1 = 0, c1 = 0, r2 = 0, c2 = 0;


	cout << "What operation would you like to perform?\n1.Addition of
Matrices\n2.Multipication of Matices\n3.Multiplication of a Vector and Matrix" << endl;
	cin >> a;

	switch (a)
	{
		case 1: cout << "Enter no. of rows followed by columns(upto 100):" << endl;
			cin >> r1 >> c1;
			r2 = r1;
			c2 = c1;
			#pragma omp parallel for
			for (int i = 0; i < r1; ++i)
			{
				for (int j = 0; j < c1; ++j)
				{
					temp >> matrixA[i][j];
					cout << matrixA[i][j] << " ";
				}
				cout << "\n";
			}
			cout << "\n";
			#pragma omp parallel for
			for (int i = 0; i < r2; ++i)
			{
				for (int j = 0; j < c2; ++j)
				{
					temp >> matrixB[i][j];
					cout << matrixB[i][j] << " ";
				}
				cout << "\n";
			}
			cout << "\n";
			#pragma omp parallel for
```

```
                        for (int i = 0; i < r1; ++i)
                        {
                                for (int j = 0; j < c1; ++j)
                                {
                                        Answer[i][j] = matrixA[i][j]+matrixB[i][j];
                                        cout << Answer[i][j] << " ";
                                }
                                cout << "\n";
                        }
                        break;

                case 2: cout << "Enter no. of rows(Matrix A)(upto 100) followed by
        columns(Matrix A)(upto 100) followed by columns(Matrix B)(upto 100):" << endl;
                        cin >> r1 >> c1 >> c2;
                        r2 = c1;

                        #pragma omp parallel for
                        for (int i = 0; i < r1; ++i)
                        {
                                for (int j = 0; j < c1; ++j)
                                {
                                        temp >> matrixA[i][j];
                                        cout << matrixA[i][j] << " ";
                                }
                                cout << "\n";
                        }
                        cout << "\n";
                        #pragma omp parallel for
                        for (int i = 0; i < r2; ++i)
                        {
                                for (int j = 0; j < c2; ++j)
                                {
                                        temp >> matrixB[i][j];
                                        cout << matrixB[i][j] << " ";
                                }
                                cout << "\n";
                        }
                        cout << "\n";

                        #pragma omp parallel for
                        for (int i = 0; i < r1; ++i)
                        {
                                for (int j = 0; j < c2; ++j)
                                {
                                        for (int k = 0; k < r2; ++k)
                                        {
                                                Answer[i][j] += matrixA[i][k] * matrixB[k][j];
                                        }
                        cout << Answer[i][j] << " ";
                    }
                    cout << "\n";
                }
                break;

            case 3: cout << "Vector into Matrix(enter 1) or Matrix into Vector(enter 2)?"
    << endl;
                int b;
                cin >> b;
                if (b == 1)
                {
                    cout << "Enter dimensions of the matrix(rows followed by
    columns):" << endl;
                    cin >> r2 >> c2;
                    r1 = 1;
                    c1 = r2;
                }
                else if (b == 2)
                {
                    cout << "Enter dimensions of the matrix(rows followed by
    columns):" << endl;
                    cin >> r1 >> c1;
                    r2 = c1;
                    c2 = 1;
                }
                else {
                    cout << "Invalid input!" << endl;
                    break;
                }

                #pragma omp parallel for
                for (int i = 0; i < r1; ++i)
                {
                    for (int j = 0; j < c1; ++j)
                    {
                        temp >> matrixA[i][j];
                        cout << matrixA[i][j] << " ";
                    }
                    cout << "\n";
                }
                cout << "\n";
                #pragma omp parallel for
                for (int i = 0; i < r2; ++i)
                {
                    for (int j = 0; j < c2; ++j)
                    {
                        temp >> matrixB[i][j];
                        cout << matrixB[i][j] << " ";
                    }
                    cout << "\n";
                }
                cout << "\n";

                #pragma omp parallel for
                for (int i = 0; i < r1; ++i)
                {
                    for (int j = 0; j < c2; ++j)
                    {
                        for (int k = 0; k < r2; ++k)
                        {
                            Answer[i][j] += matrixA[i][k] * matrixB[k][j];
                        }
                        cout << Answer[i][j] << " ";
                    }
                    cout << "\n";
                }
                break;

            default: cout << "Invalid Input!" << endl;
                break;
        }

        temp.close();

        return 0;
}
```

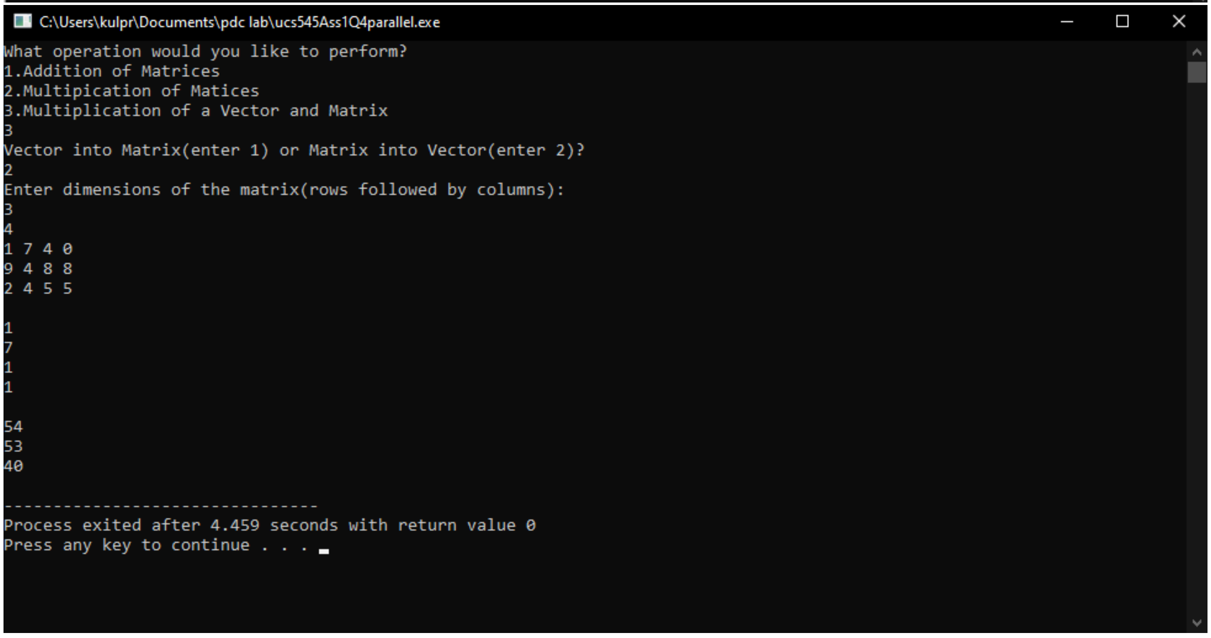


```
C:\Users\kulpr\Documents\pdc lab\ucs545Ass1Q4parallel.exe
What operation would you like to perform?
1.Addition of Matrices
2.Multipication of Matices
3.Multiplication of a Vector and Matrix
1
Enter no. of rows followed by columns(upto 100):
4
5
1 7 4 0 9
4 8 8 2 4
5 5 1 7 1
1 5 2 7 6

1 4 2 3 2
2 1 6 8 5
7 6 1 8 9
2 7 9 5 4

2 11 6 3 11
6 9 14 10 9
12 11 2 15 10
3 12 11 12 10

--------------------------------
Process exited after 4.652 seconds with return value 0
Press any key to continue . . .
```



C:\Users\kulpr\Documents\pdc lab\ucs545Ass1Q4parallel.exe

```
What operation would you like to perform?
1.Addition of Matrices
2.Multipication of Matices
3.Multiplication of a Vector and Matrix
2
Enter no. of rows(Matrix A)(upto 100) followed by columns(Matrix A)(upto 100) followed by columns(Matrix B)(upto 100):
2
2
6
1 7
4 0

9 4 8 8 2 4
5 5 1 7 1 1

44 39 15 57 9 11
36 16 32 32 8 16

--------------------------------
Process exited after 7.514 seconds with return value 0
Press any key to continue . . .
```



C:\Users\kulpr\Documents\pdc lab\ucs545Ass1Q4parallel.exe

```
What operation would you like to perform?
1.Addition of Matrices
2.Multipication of Matices
3.Multiplication of a Vector and Matrix
3
Vector into Matrix(enter 1) or Matrix into Vector(enter 2)?
2
Enter dimensions of the matrix(rows followed by columns):
3
4
1 7 4 0
9 4 8 8
2 4 5 5

1
7
1
1

54
53
40

--------------------------------
Process exited after 4.459 seconds with return value 0
Press any key to continue . . .
```