OS-Lab

Lab#11

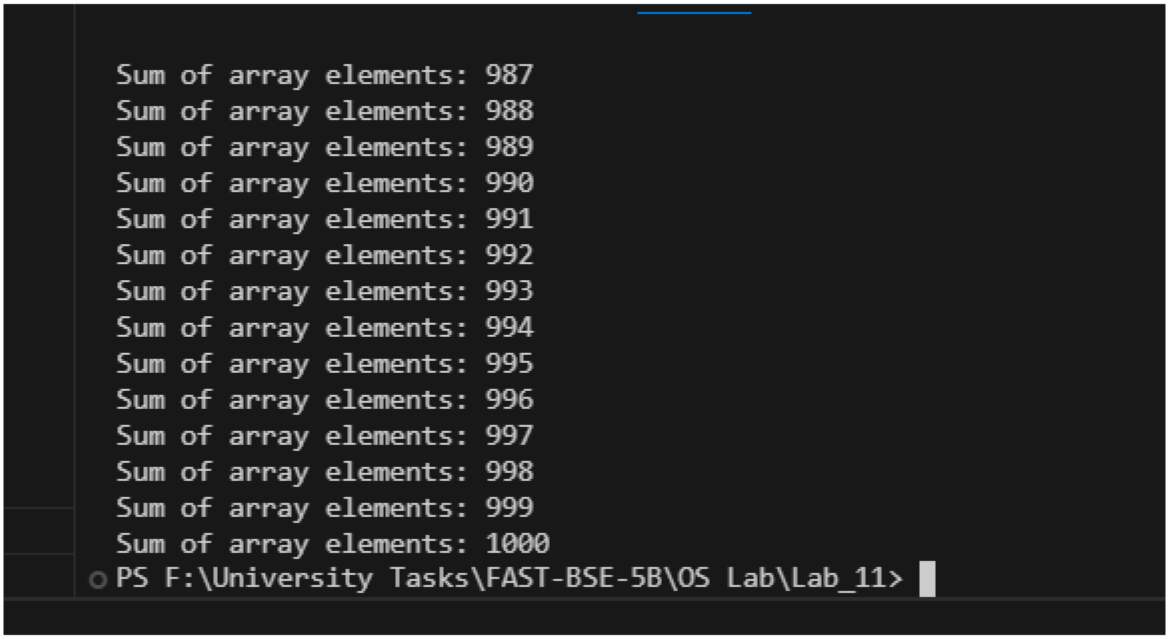
Name:Anas Altaf

Roll.No: 22F-3639

Codes:

T-1:

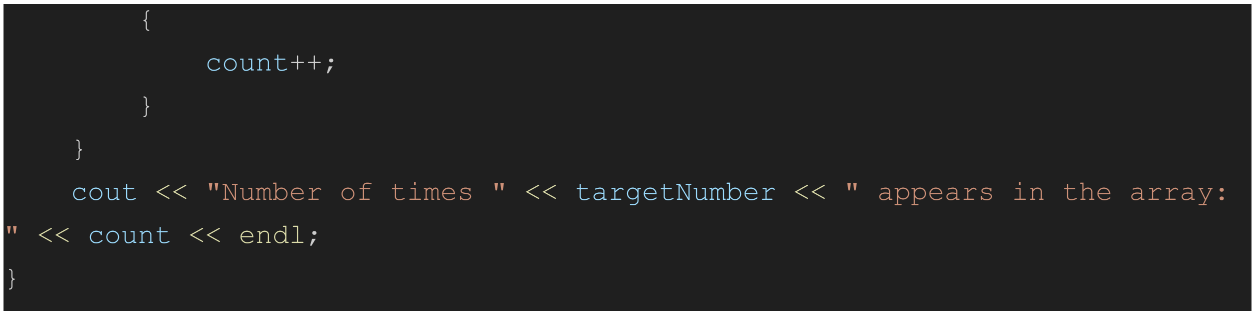
```
#include <iostream>
#include <omp.h>
#include <vector>
using namespace std;

int main()
{

    const int N = 1000;

    // Initialize an array of size N with all elements as 1
    vector<int> array(N, 1);

    int sum = 0;

    // Parallelize the for loop

#pragma omp parallel for reduction(+ : sum)

    for (int i = 0; i < N; i++)
    {
        printf("Sum of array elements: %d\n", sum);
        sum += array[i];
    }
    std::cout << "Sum of array elements: " << sum << std::endl;

    return 0;
}
```



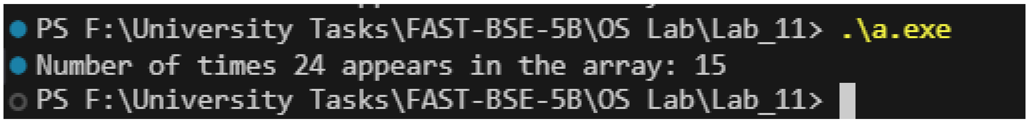
```
Sum of array elements: 987
Sum of array elements: 988
Sum of array elements: 989
Sum of array elements: 990
Sum of array elements: 991
Sum of array elements: 992
Sum of array elements: 993
Sum of array elements: 994
Sum of array elements: 995
Sum of array elements: 996
Sum of array elements: 997
Sum of array elements: 998
Sum of array elements: 999
Sum of array elements: 1000
PS F:\University Tasks\FAST-BSE-5B\OS Lab\Lab_11>
```

T-2:

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

int main()
{
    int count = 0;
    int targetNumber = 24;
    int arr[1000];
    for (int i = 0; i < 1000; i++)
    {
        arr[i] = rand() % 100;
    }
#pragma omp parallel for reduction(+ : count)
    for (int i = 0; i < 1000; i++)
    {
        if (arr[i] == targetNumber)
```

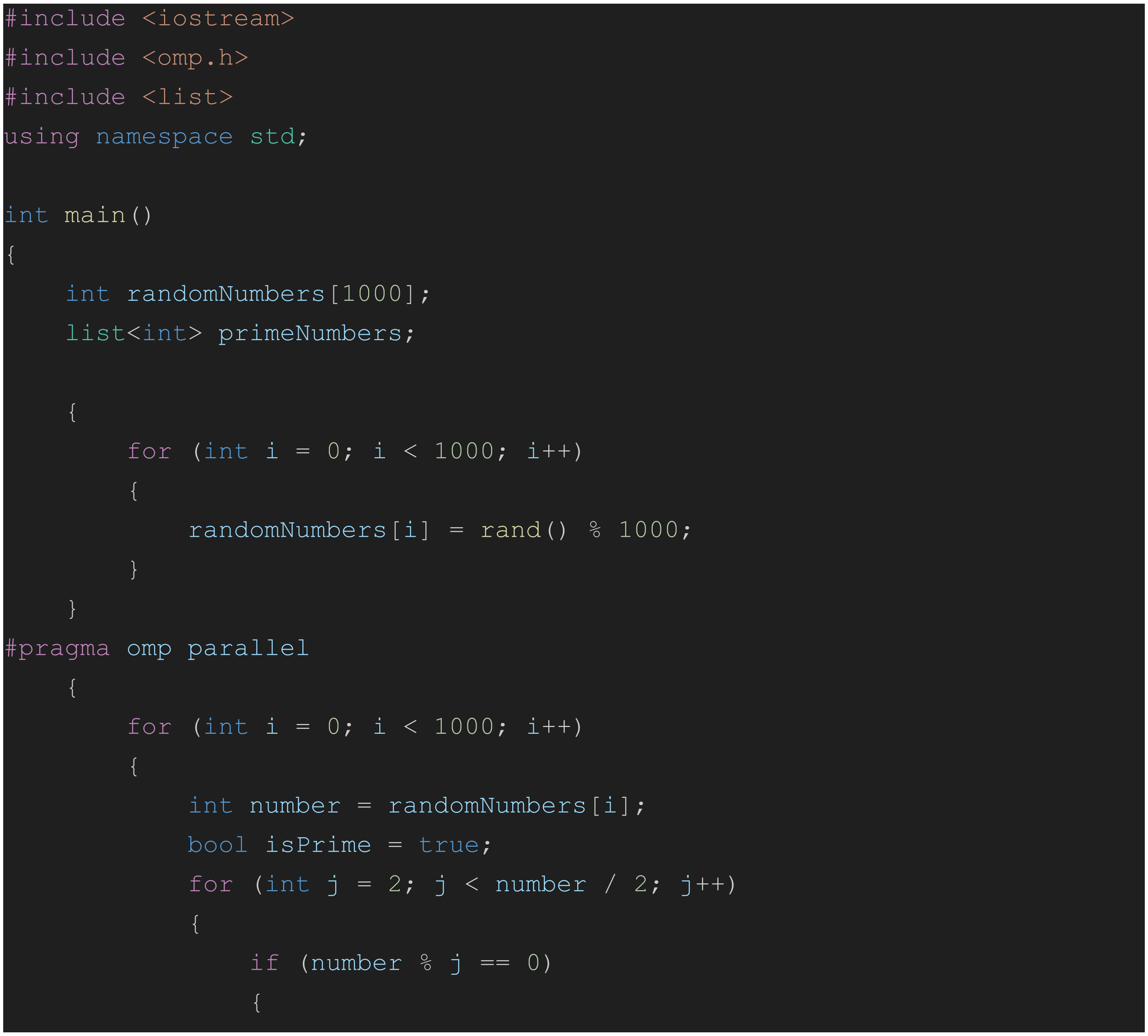


```
        {
            count++;
        }
    }
    cout << "Number of times " << targetNumber << " appears in the array:
" << count << endl;
}
```

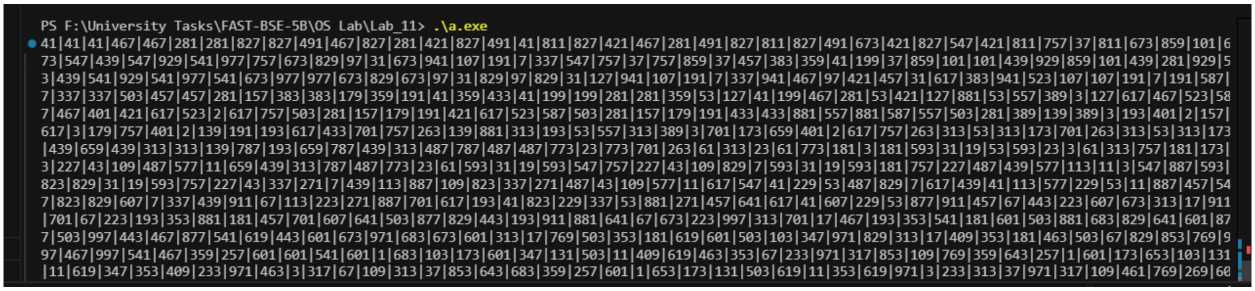


```
PS F:\University Tasks\FAST-BSE-5B\OS Lab\Lab_11> .\a.exe
Number of times 24 appears in the array: 15
PS F:\University Tasks\FAST-BSE-5B\OS Lab\Lab_11>
```

T-3:



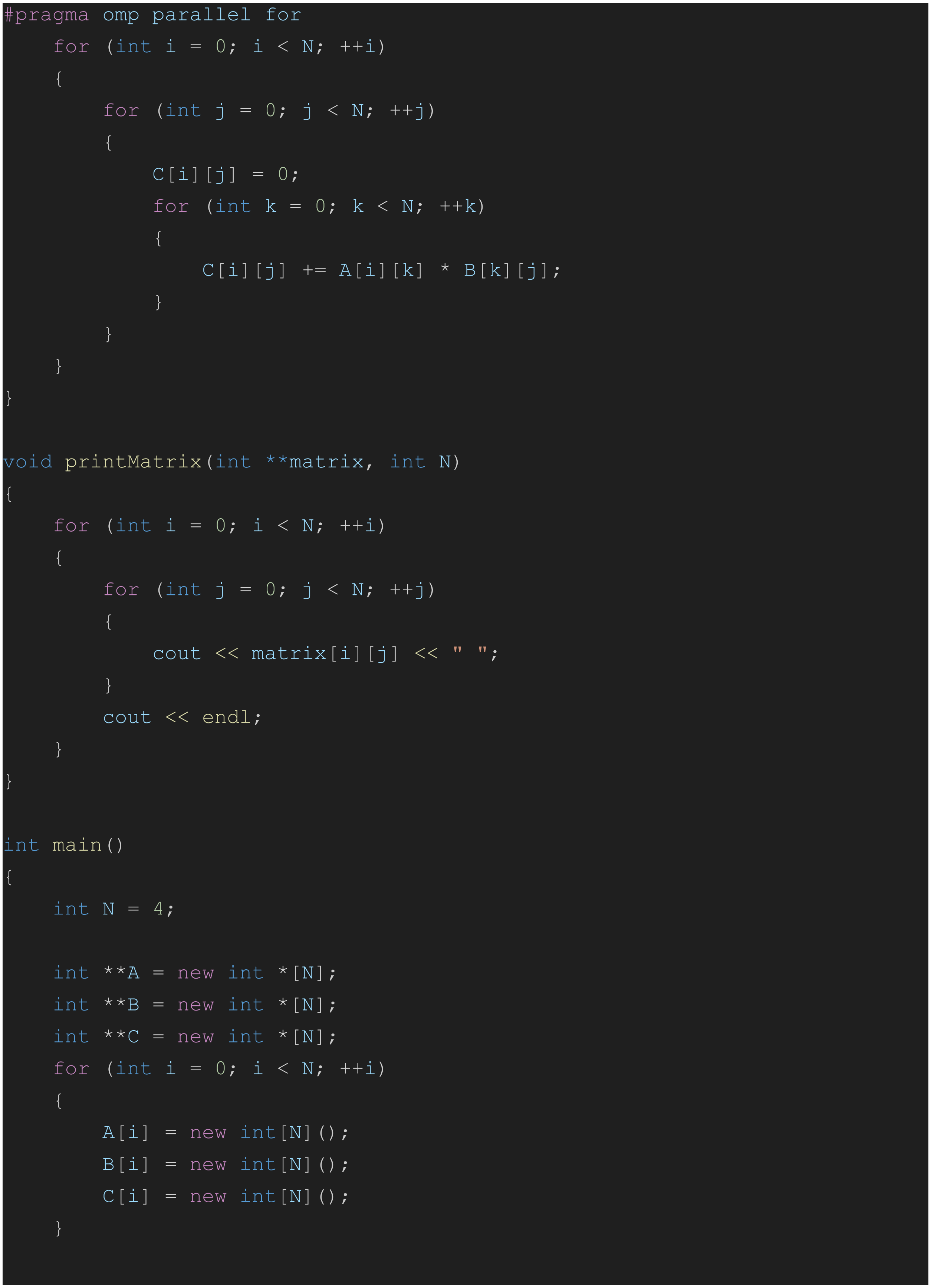
```
#include <iostream>
#include <omp.h>
#include <list>
using namespace std;

int main()
{
    int randomNumbers[1000];
    list<int> primeNumbers;

    {
        for (int i = 0; i < 1000; i++)
        {
            randomNumbers[i] = rand() % 1000;
        }
    }
#pragma omp parallel
    {
        for (int i = 0; i < 1000; i++)
        {
            int number = randomNumbers[i];
            bool isPrime = true;
            for (int j = 2; j < number / 2; j++)
            {
                if (number % j == 0)
                {
                    isPrime = false;
                    break;
                }
            }
            if (isPrime)
            {

#pragma omp critical
                {
                    primeNumbers.push_back(number);
                }
            }
        }
    }

    for (const auto &number : primeNumbers)
    {
        cout << number << "|";
    }
    return 0;
}
```



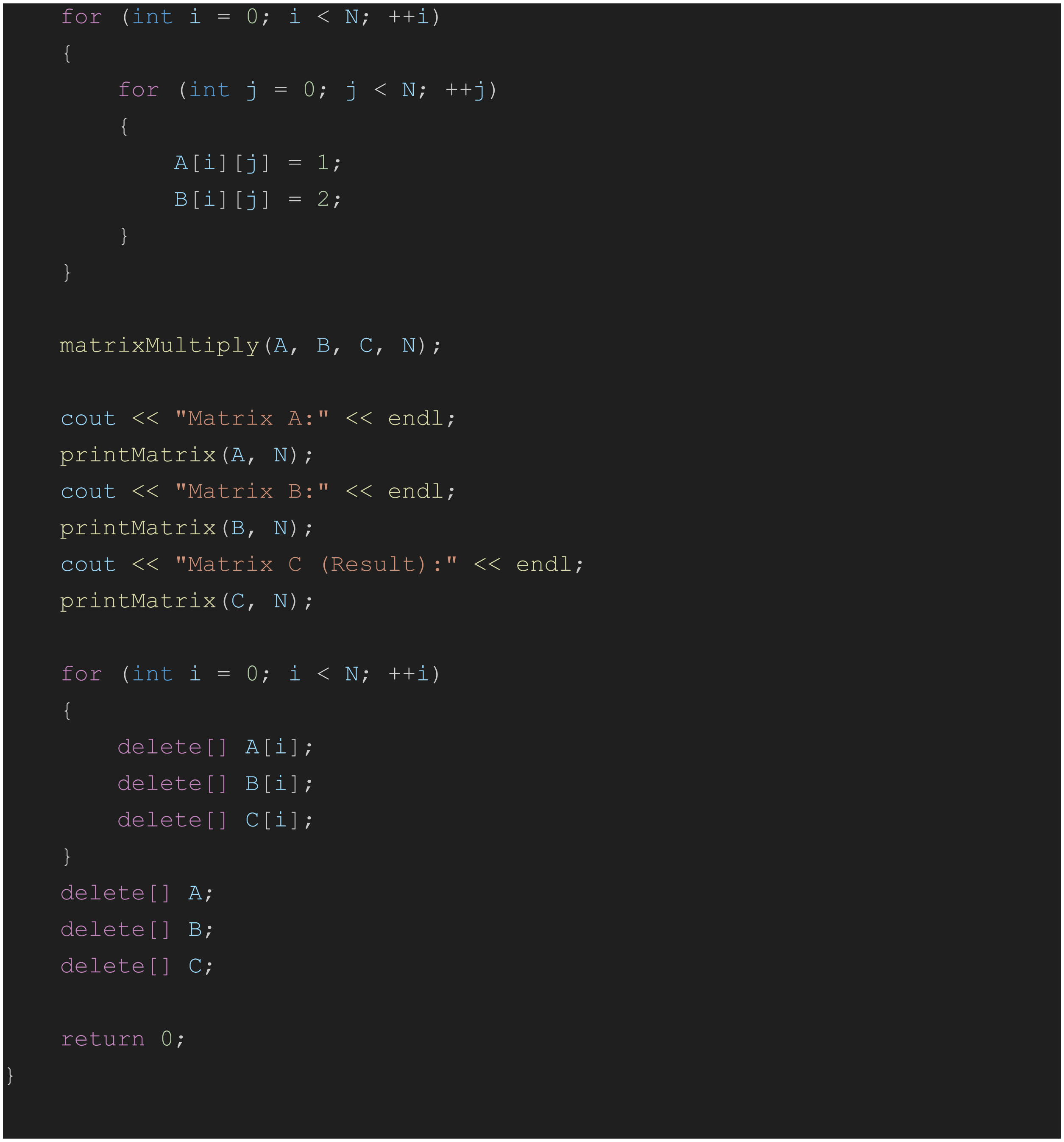
```
PS F:\University Tasks\FAST-BSE-5B\OS Lab\Lab_11> .\a.exe
41|41|41|467|467|281|281|827|827|491|467|827|281|421|827|491|41|811|827|421|467|281|491|827|811|827|491|673|421|827|547|421|811|757|37|811|673|859|101|6
73|547|439|547|929|541|977|757|673|829|97|31|673|941|107|191|7|337|547|757|37|757|859|37|457|383|359|41|199|37|859|101|101|439|929|859|101|439|281|929|5
3|439|541|929|541|977|541|673|977|977|673|829|673|97|31|829|97|829|31|127|941|107|191|7|337|941|467|97|421|457|31|617|383|941|523|107|107|191|7|191|587|
7|337|337|503|457|457|281|157|383|383|179|359|191|41|359|433|41|199|199|281|281|359|53|127|41|199|467|281|53|421|127|881|53|557|389|3|127|617|467|523|58
7|467|401|421|617|523|2|617|757|503|281|157|179|191|421|617|523|587|503|281|157|179|191|433|433|881|557|881|587|557|503|281|389|139|389|3|193|401|2|157|
617|3|179|757|401|2|139|191|193|617|433|701|757|263|139|881|313|193|53|557|313|389|3|701|173|659|401|2|617|757|263|313|53|313|173|701|263|313|53|313|173
|439|659|439|313|313|139|787|193|659|787|439|313|487|787|487|487|773|23|773|701|263|61|313|23|61|773|181|3|181|593|31|19|53|593|23|3|61|313|757|181|173|
3|227|43|109|487|577|11|659|439|313|787|487|773|23|61|593|31|19|593|547|757|227|43|109|829|7|593|31|19|593|181|757|227|487|439|577|113|11|3|547|887|593|
823|829|31|19|593|757|227|43|337|271|7|439|113|887|109|823|337|271|487|43|109|577|11|617|547|41|229|53|487|829|7|617|439|41|113|577|229|53|11|887|457|54
7|823|829|607|7|337|439|911|67|113|223|271|887|701|617|193|41|823|229|337|53|881|271|457|641|617|41|607|229|53|877|911|457|67|443|223|607|673|313|17|911
|701|67|223|193|353|881|181|457|701|607|641|503|877|829|443|193|911|881|641|67|673|223|997|313|701|17|467|193|353|541|181|601|503|881|683|829|641|601|87
7|503|997|443|467|877|541|619|443|601|673|971|683|673|601|313|17|769|503|353|181|619|601|503|103|347|971|829|313|17|409|353|181|463|503|67|829|853|769|9
97|467|997|541|467|359|257|601|601|541|601|1|683|103|173|601|347|131|503|11|409|619|463|353|67|233|971|317|853|109|769|359|643|257|1|601|173|653|103|131
|11|619|347|353|409|233|971|463|3|317|67|109|313|37|853|643|683|359|257|601|1|653|173|131|503|619|11|353|619|971|3|233|313|37|971|317|109|461|769|269|60
```

T-4:

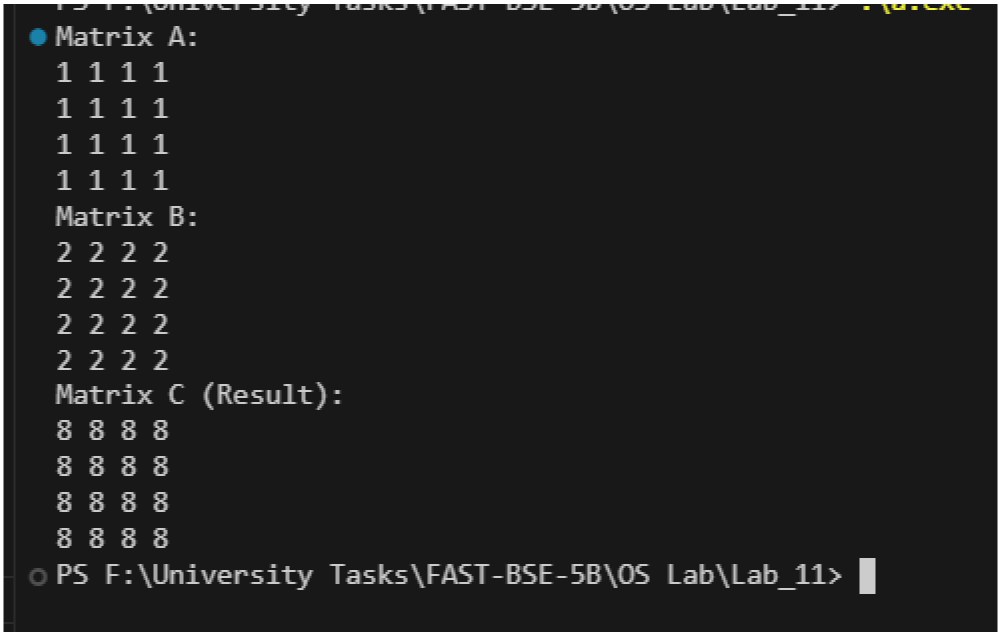
```
#include <iostream>
#include <omp.h>

using namespace std;

void matrixMultiply(int **A, int **B, int **C, int N)
{
```



```cpp
#pragma omp parallel for
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {
            C[i][j] = 0;
            for (int k = 0; k < N; ++k)
            {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

void printMatrix(int **matrix, int N)
{
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main()
{
    int N = 4;

    int **A = new int *[N];
    int **B = new int *[N];
    int **C = new int *[N];
    for (int i = 0; i < N; ++i)
    {
        A[i] = new int[N]();
        B[i] = new int[N]();
        C[i] = new int[N]();
    }
```



```
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
        {
            A[i][j] = 1;
            B[i][j] = 2;
        }
    }

    matrixMultiply(A, B, C, N);

    cout << "Matrix A:" << endl;
    printMatrix(A, N);
    cout << "Matrix B:" << endl;
    printMatrix(B, N);
    cout << "Matrix C (Result):" << endl;
    printMatrix(C, N);

    for (int i = 0; i < N; ++i)
    {
        delete[] A[i];
        delete[] B[i];
        delete[] C[i];
    }
    delete[] A;
    delete[] B;
    delete[] C;

    return 0;
}
```



```
PS F:\University Tasks\FAST-BSE-5B\OS Lab\Lab_11> .\a.exe
Matrix A:
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
Matrix B:
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
Matrix C (Result):
8 8 8 8
8 8 8 8
8 8 8 8
8 8 8 8
PS F:\University Tasks\FAST-BSE-5B\OS Lab\Lab_11>
```