Nikhil Rajendra Danapgol
HPC Assignment 2
2019BTECCS00036

Program for Vector to Vector Addition

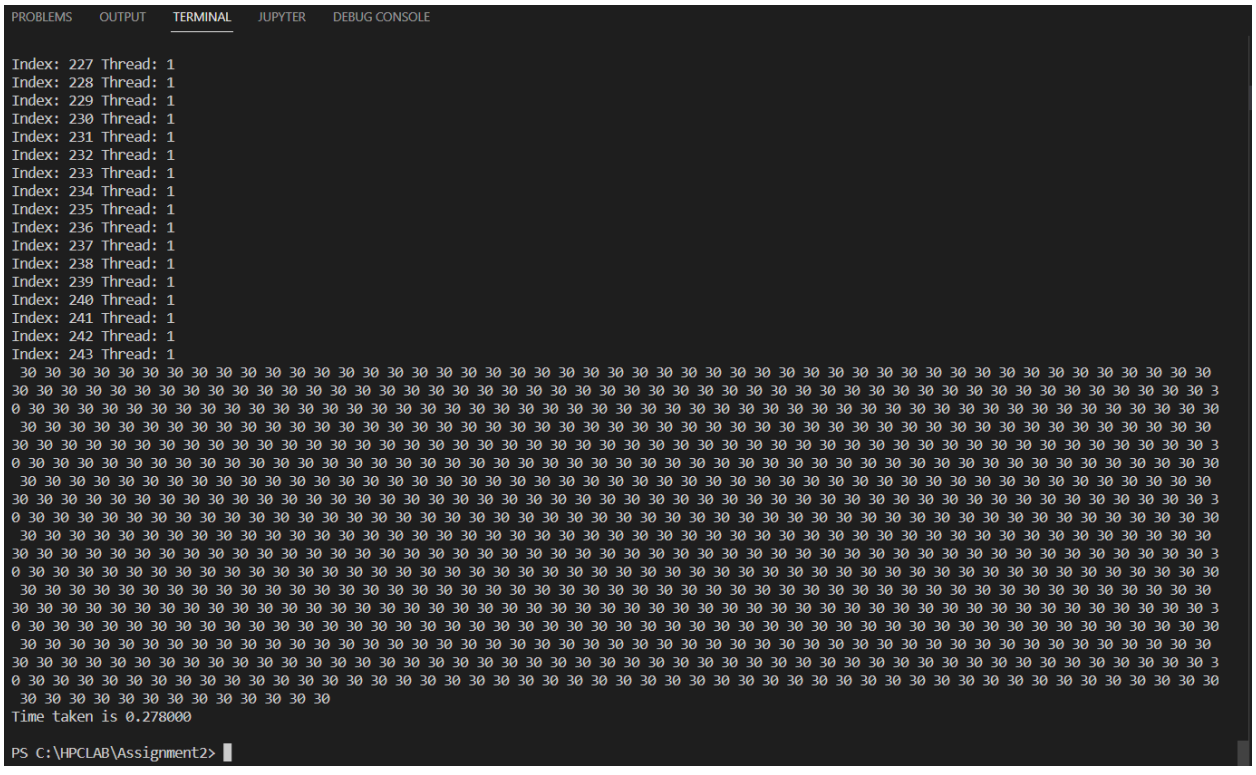
vectorAdditionPara.cpp

vectorAdditionPara.cpp > main()

```cpp
#include <omp.h>
#include <pthread.h>
#include <stdio.h>

int main()
{
    int N = 1000;
    int mat1[1000];
    for (int i = 0; i < N; i++)
        mat1[i] = 10;

    int mat2[1000];
    for (int i = 0; i < N; i++)
        mat2[i] = 20;

    int ans[1000] = {0};
    double stime = omp_get_wtime();
#pragma omp parallel for reduction(+ \
                                   : ans)
    for (int i = 0; i < N; i++)
    {
        ans[i] = mat1[i] + mat2[i];
        printf("Index: %d Thread: %d\n", i, omp_get_thread_num());
    }
```

```
    for (int i = 0; i < N; i++)
    {
        printf("%d ", ans[i]);
    }

    double etime = omp_get_wtime();
    double time = etime - stime;
    printf("\nTime taken is %f\n", time);
    printf("\n");
    return 0;
}
```

Output:



Sequential Algorithm:

vectorAdditionSeq.cpp > ...

```cpp
#include <omp.h>
#include <pthread.h>
#include <stdio.h>

int main()
{
    int N = 1000;
    int mat1[1000];
    for (int i = 0; i < N; i++)
        mat1[i] = 10;

    int mat2[1000];
    for (int i = 0; i < N; i++)
        mat2[i] = 20;

    int ans[1000] = {0};
    double stime = omp_get_wtime();
    for (int i = 0; i < N; i++)
    {
        ans[i] = mat1[i] + mat2[i];
        printf("Index: %d Thread: %d\n", i, omp_get_thread_num());
    }

    for (int i = 0; i < N; i++)
    {
        printf("%d ", ans[i]);
    }

    double etime = omp_get_wtime();
    double time = etime - stime;
    printf("Time taken is %f\n", time);
    return 0;
}
```

Output:

```
Index: 995 Thread: 0
Index: 996 Thread: 0
Index: 997 Thread: 0
Index: 998 Thread: 0
Index: 999 Thread: 0
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3
0 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
 30 30 30 30 30 30 30 30 30 30 30 30 30
Time taken is 0.259000
PS C:\HPCLAB\Assignment2>
```

Program for Vector-Scalar Multiplication:

Sequential:

vectorScalarMultiplicationSeq.cpp > main()

```cpp
#include <omp.h>
#include <pthread.h>
#include <stdio.h>

int main()
{
    int N = 500;
    int mat1[501];
    for (int i = 0; i < N; i++)
        mat1[i] = 10;
    int S = 6;

    double itime;
    itime = omp_get_wtime();
    for (int i = 0; i < N; i++)
    {
        mat1[i] *= S;
        printf("Index: %d Thread: %d\n", i, omp_get_thread_num());
    }

    for (int i = 0; i < N; i++)
    {
        printf("%d ", mat1[i]);
    }

    double ftime = omp_get_wtime();
    double exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
}

```

Output:

```
Index: 494 Thread: 0
Index: 495 Thread: 0
Index: 496 Thread: 0
Index: 497 Thread: 0
Index: 498 Thread: 0
Index: 499 Thread: 0
60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 6
0 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 6
0 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 6
0 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 6
0 60 60 60 60 60 60
Time taken is 0.111000

PS C:\HPCLAB\Assignment2>
```

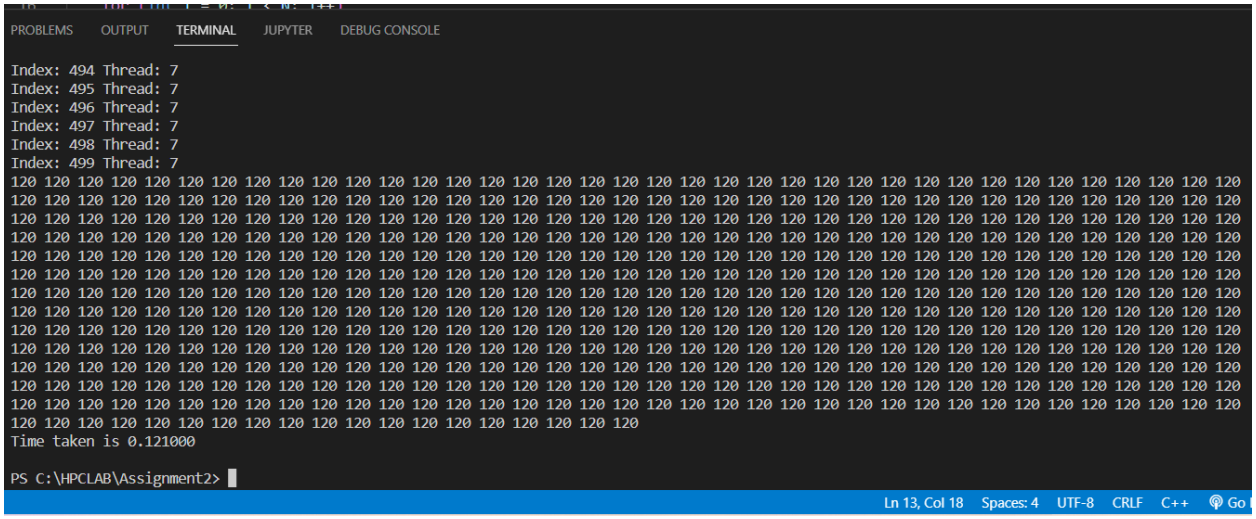
Ln 24, Col 6 Spaces: 4 UTF-8 CRLF C++ Go Liv

Parallel :

vectorScalarMultiplicationPara.cpp > main()

```cpp
#include <omp.h>
#include <pthread.h>
#include <stdio.h>

int main()
{
    int N = 500;
    int mat1[501];
    for (int i = 0; i < N; i++)
        mat1[i] = 20;
    int S = 6;

    double itime;
    itime = omp_get_wtime();
#pragma omp parallel for
    for (int i = 0; i < N; i++)
    {
        mat1[i] *= S;
        printf("Index: %d Thread: %d\n", i, omp_get_thread_num());
    }

    for (int i = 0; i < N; i++)
    {
        printf("%d ", mat1[i]);
    }

    double ftime = omp_get_wtime();
    double exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
}
```

Output:



Github Link:

https://github.com/nd22052000/HPC