Final Year B. Tech, Sem VII 2022-23
PRN – 2020BTECS00211
Name – Aashita Narendra Gupta
High Performance Computing Lab
Batch: B4

Practical no - 2

Github Link for Code - https://github.com/Aashita06/HPC_Practicals

❖ SPMD: Single Program Multiple Data

→

It is a parallel programming style. In SPMD style tasks are split up and run simultaneously on multiple processors with different data. This is done to achieve results faster.

❖ Worksharing

→

Threads are assigned an independent subset of the total workload For example, different chunks of an iteration are distributed among the threads.
Eg.
T1 -> iteration from 1 to 5
T2 -> iteration from 6 to 10 and so on…

OpenMP provides various constructs for worksharing. OpenMP loop worksharing construct OpenMP's loop worksharing construct splits loop iterations among all active threads.
#pragma omp for.

❖ Types of variables

→

1. Shared Variables:
- There exists one instance of this variable which is shared among all threads.

2. Private Variables:
- Each thread in a team of threads has its own local copy of the private variable.

Two ways to assign variables as private and shared are Implicit and Explicit.

Implicit: All the variables declared outside of the pragma are by default shared and all the variables declared inside pragma are private.

Explicit:

- **Shared Clause**

eg. #pragma omp parallel for shared(n, a) => n and a are declared as shared variables.

- **Private Clause**

eg. #pragma omp parallel for shared(n, a) private(c) => here c is private variable.

- **Default Clause**

eg. #pragma omp parallel for default(shared) => now all variables are shared
#pragma omp parallel for default(private) => now all variables are private.

firstprivate

- firstprivate make the variable private but that variable is initialised with the value that it has before the parallel region.

lastprivate

- lastprivate make the variable private but it retain the last value of that private variable outside of the private region.

❖ **Schedule**

- a specification of how iterations of associated loops are divided into contiguous non-empty subsets.

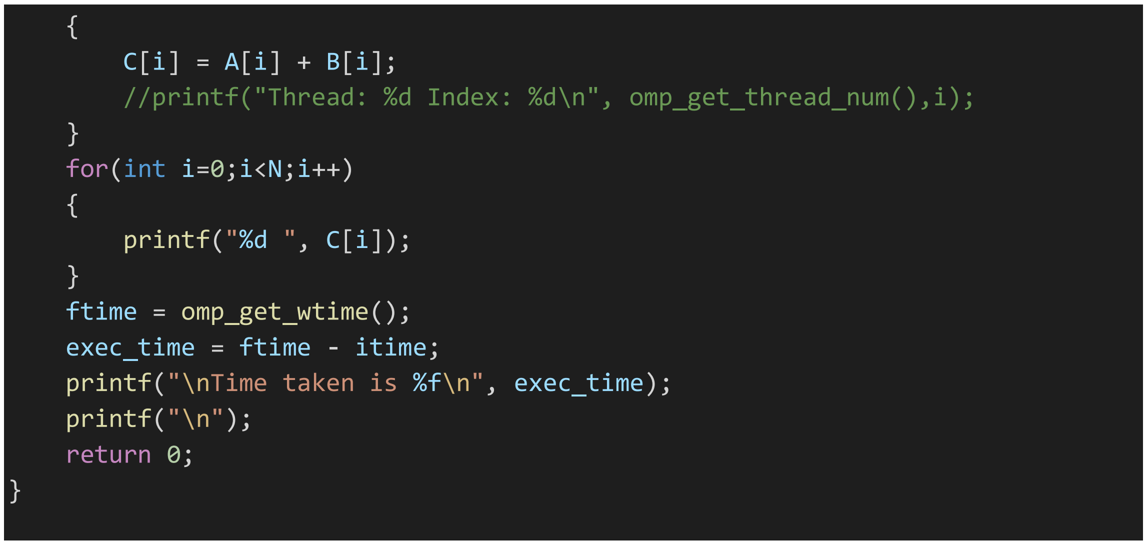
syntax: #pragma omp parallel for schedule([modifier [modifier]:]kind[,chunk_size])

❖ **Program for Vector-to-Vector Addition**

→

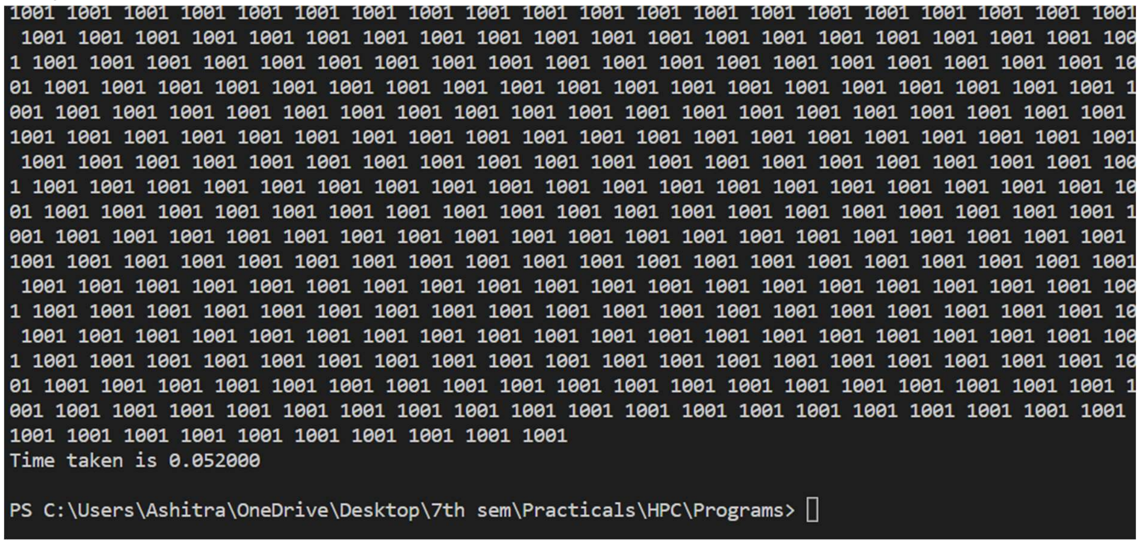
Serial Code:

```
#include <omp.h>
#include <stdio.h>
#include <pthread.h>
int main()
{
    int N = 1000;
    int A[1000];
    for(int i=0;i<N;i++)A[i] = i + 1;
        int B[1000];

    for(int i=0;i<N;i++)B[i] = N - i;
        int C[1000] = {0};

    double itime, ftime, exec_time;
    itime = omp_get_wtime();

    for (int i = 0; i < N; i++)
```



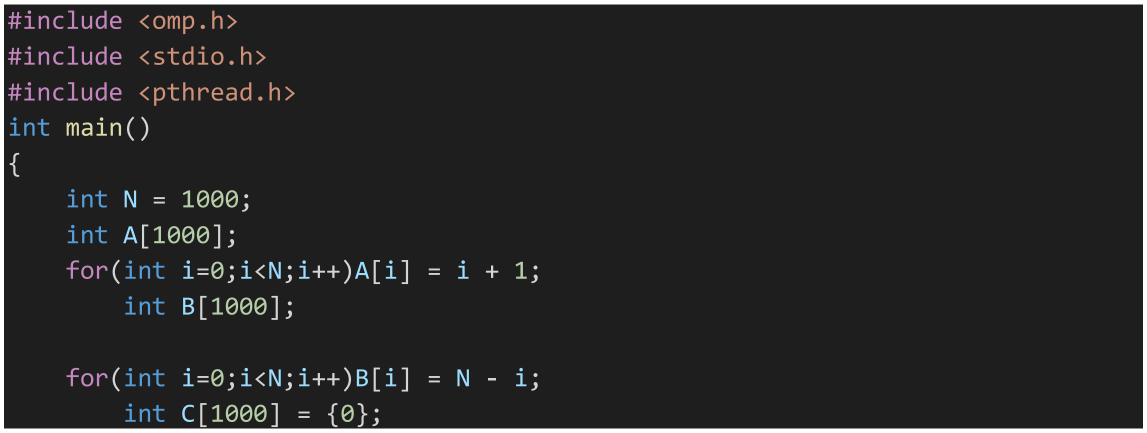
```c
    {
        C[i] = A[i] + B[i];
        //printf("Thread: %d Index: %d\n", omp_get_thread_num(),i);
    }
    for(int i=0;i<N;i++)
    {
        printf("%d ", C[i]);
    }
    ftime = omp_get_wtime();
    exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
    return 0;
}
```

Output:

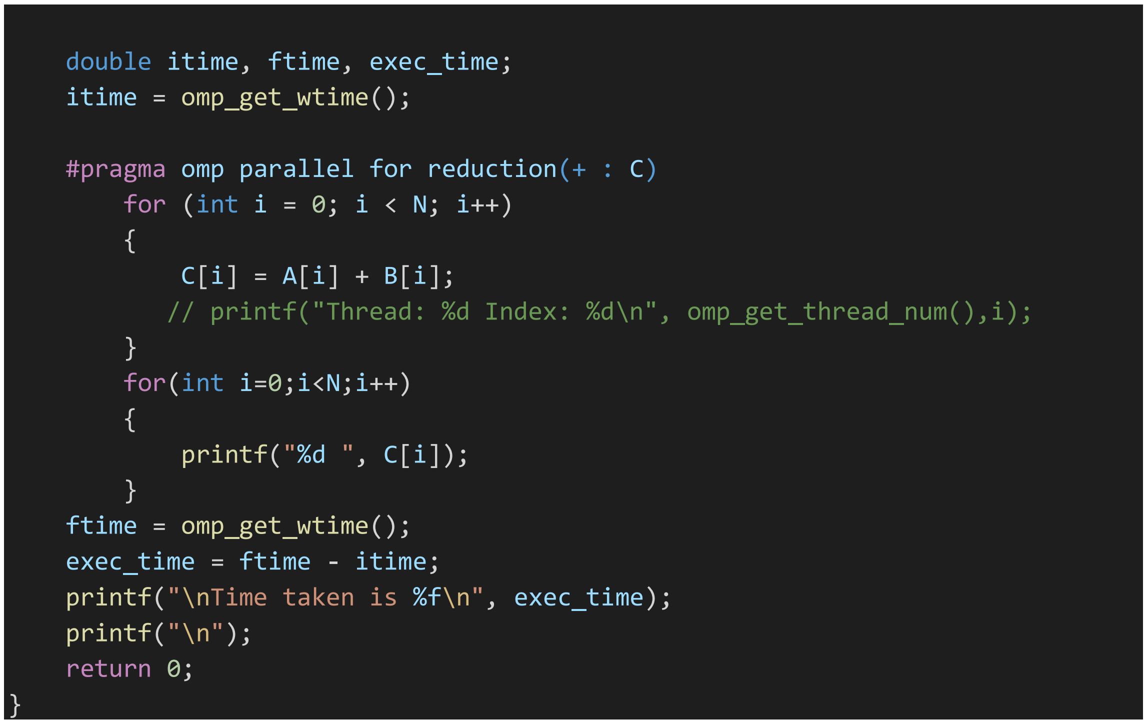


```
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
Time taken is 0.052000

PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> 
```

Parallel Code:

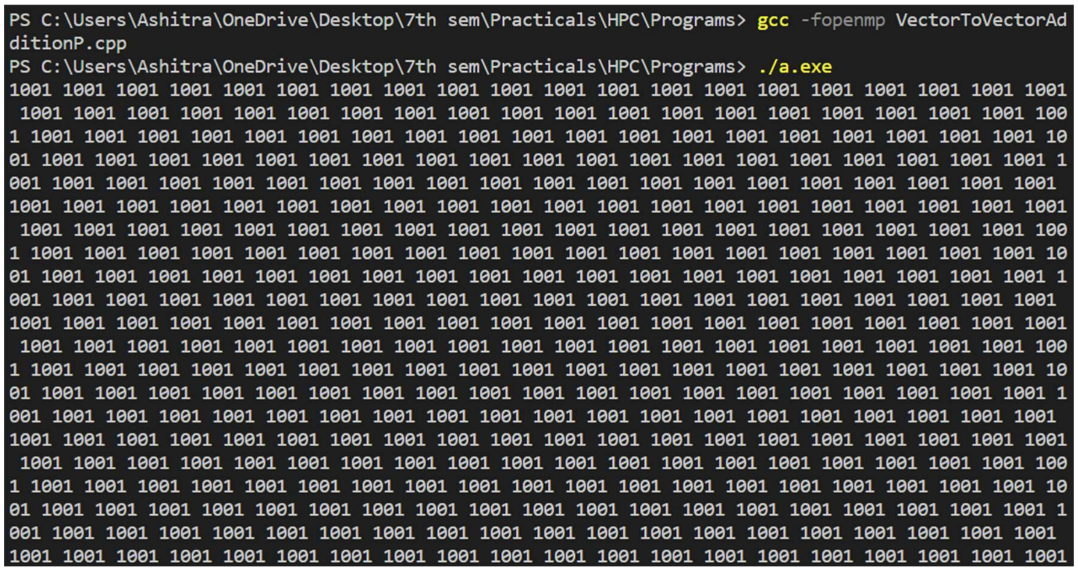


```c
#include <omp.h>
#include <stdio.h>
#include <pthread.h>
int main()
{
    int N = 1000;
    int A[1000];
    for(int i=0;i<N;i++)A[i] = i + 1;
        int B[1000];

    for(int i=0;i<N;i++)B[i] = N - i;
        int C[1000] = {0};
```

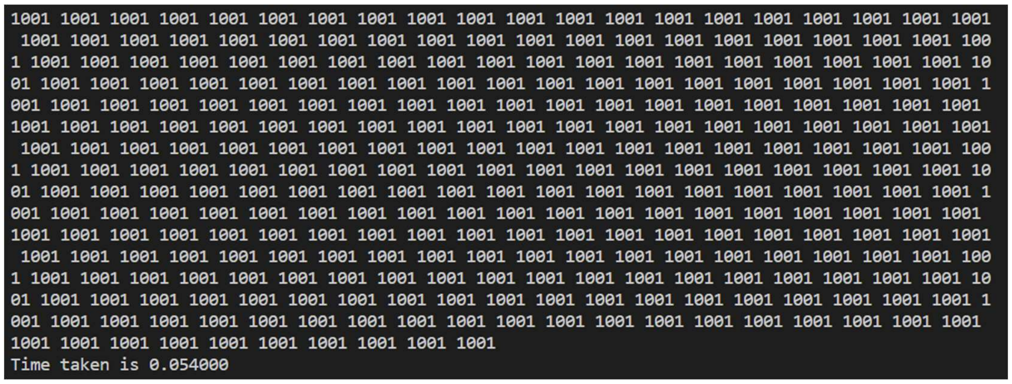


```
    double itime, ftime, exec_time;
    itime = omp_get_wtime();

    #pragma omp parallel for reduction(+ : C)
        for (int i = 0; i < N; i++)
        {
            C[i] = A[i] + B[i];
           // printf("Thread: %d Index: %d\n", omp_get_thread_num(),i);
        }
        for(int i=0;i<N;i++)
        {
            printf("%d ", C[i]);
        }
    ftime = omp_get_wtime();
    exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
    return 0;
}
```

Output:



```
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> gcc -fopenmp VectorToVectorAd
ditionP.cpp
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> ./a.exe
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
```



```
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 100
1 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 10
01 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1
001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
1001 1001 1001 1001 1001 1001 1001 1001 1001 1001
Time taken is 0.054000
```

Speedup = Ts/Tp = 0.05200/0.05400 = 0.9629

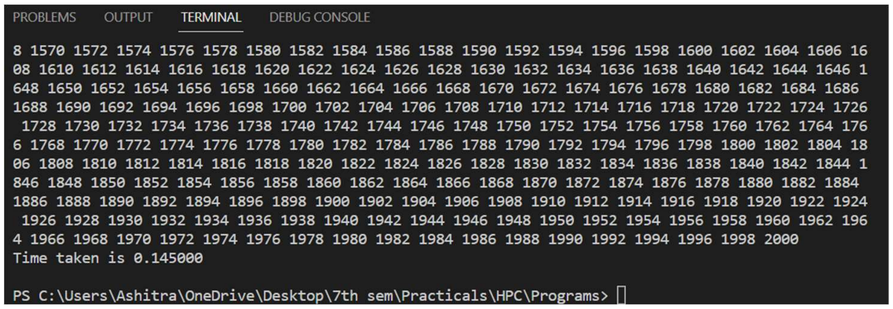
❖ **Program for Vector-Scalar Multiplication:**

→

Serial Code:

```c
#include <omp.h>
#include <stdio.h>
#include <pthread.h>
int main()
{
    int N = 1000;
    int A[1000];

    for(int i=0;i<N;i++)A[i] = i + 1;
        int S = 2;

    double itime, ftime, exec_time;
    itime = omp_get_wtime();

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

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

    ftime = omp_get_wtime();
    exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
    return 0;
}
```

Output:



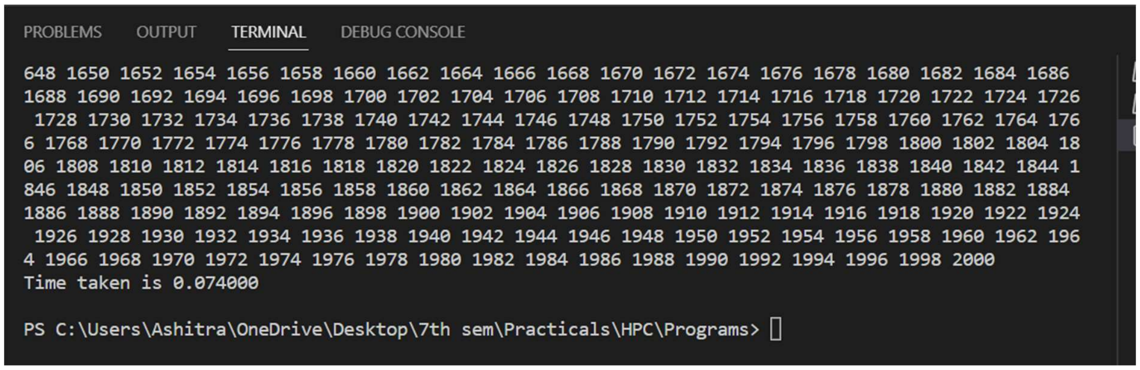
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
8 1570 1572 1574 1576 1578 1580 1582 1584 1586 1588 1590 1592 1594 1596 1598 1600 1602 1604 1606 16
08 1610 1612 1614 1616 1618 1620 1622 1624 1626 1628 1630 1632 1634 1636 1638 1640 1642 1644 1646 1
648 1650 1652 1654 1656 1658 1660 1662 1664 1666 1668 1670 1672 1674 1676 1678 1680 1682 1684 1686
1688 1690 1692 1694 1696 1698 1700 1702 1704 1706 1708 1710 1712 1714 1716 1718 1720 1722 1724 1726
 1728 1730 1732 1734 1736 1738 1740 1742 1744 1746 1748 1750 1752 1754 1756 1758 1760 1762 1764 176
6 1768 1770 1772 1774 1776 1778 1780 1782 1784 1786 1788 1790 1792 1794 1796 1798 1800 1802 1804 18
06 1808 1810 1812 1814 1816 1818 1820 1822 1824 1826 1828 1830 1832 1834 1836 1838 1840 1842 1844 1
846 1848 1850 1852 1854 1856 1858 1860 1862 1864 1866 1868 1870 1872 1874 1876 1878 1880 1882 1884
1886 1888 1890 1892 1894 1896 1898 1900 1902 1904 1906 1908 1910 1912 1914 1916 1918 1920 1922 1924
 1926 1928 1930 1932 1934 1936 1938 1940 1942 1944 1946 1948 1950 1952 1954 1956 1958 1960 1962 196
4 1966 1968 1970 1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000
Time taken is 0.145000

PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> 
```

Parallel Code:

```c
#include <omp.h>
#include <stdio.h>
#include <pthread.h>
int main()
{
    int N = 1000;
    int A[1000];

    for(int i=0;i<N;i++)A[i] = i + 1;
        int S = 2;

    double itime, ftime, exec_time;
    itime = omp_get_wtime();

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

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

    ftime = omp_get_wtime();
    exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
    return 0;
}
```

Output:


PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
648 1650 1652 1654 1656 1658 1660 1662 1664 1666 1668 1670 1672 1674 1676 1678 1680 1682 1684 1686
1688 1690 1692 1694 1696 1698 1700 1702 1704 1706 1708 1710 1712 1714 1716 1718 1720 1722 1724 1726
 1728 1730 1732 1734 1736 1738 1740 1742 1744 1746 1748 1750 1752 1754 1756 1758 1760 1762 1764 176
6 1768 1770 1772 1774 1776 1778 1780 1782 1784 1786 1788 1790 1792 1794 1796 1798 1800 1802 1804 18
06 1808 1810 1812 1814 1816 1818 1820 1822 1824 1826 1828 1830 1832 1834 1836 1838 1840 1842 1844 1
846 1848 1850 1852 1854 1856 1858 1860 1862 1864 1866 1868 1870 1872 1874 1876 1878 1880 1882 1884
1886 1888 1890 1892 1894 1896 1898 1900 1902 1904 1906 1908 1910 1912 1914 1916 1918 1920 1922 1924
 1926 1928 1930 1932 1934 1936 1938 1940 1942 1944 1946 1948 1950 1952 1954 1956 1958 1960 1962 196
4 1966 1968 1970 1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000
Time taken is 0.074000

PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs>
```

Speedup = Ts/Tp = 1.9819