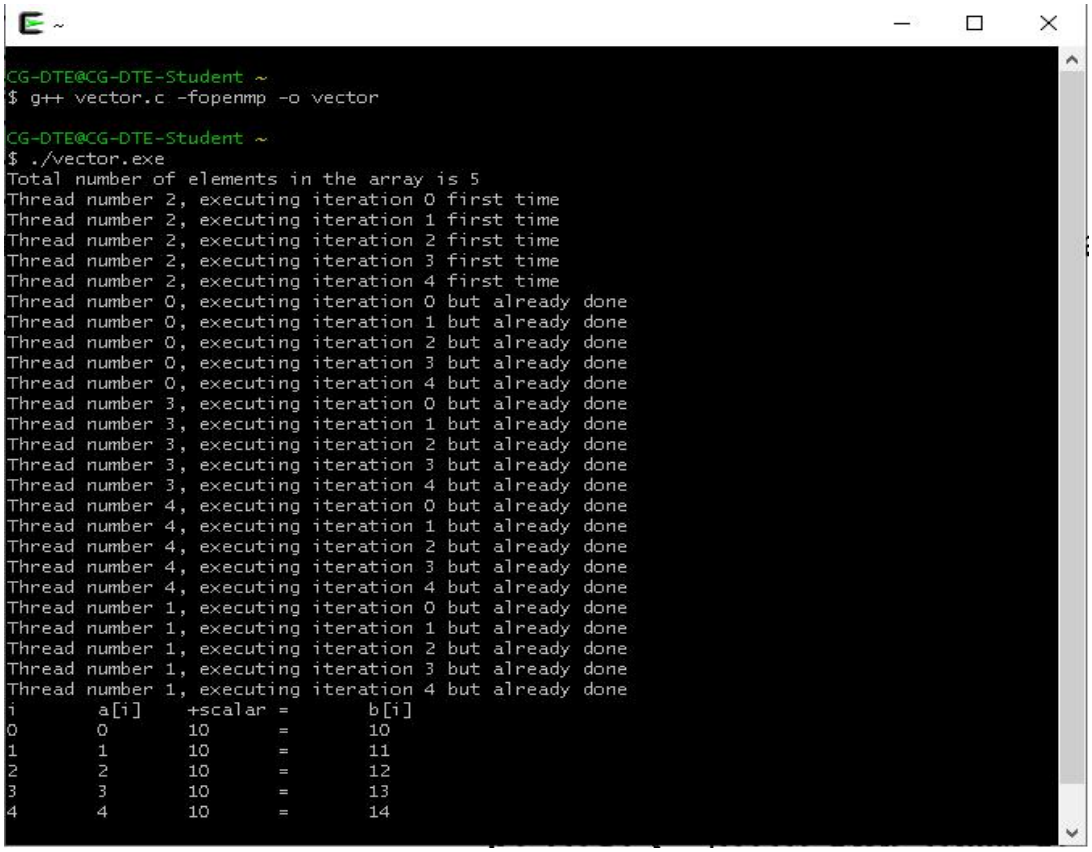
Walchand College of Engineering, Sangli Department of Computer Science
and Engineering
Class: Final Year (Computer Science and Engineering)
Year: 2021-22 Semester: 1
Course: High Performance Computing Lab
Practical No. 2
Exam Seat No : 2019BTECS00205
Name: Shweta Nandkumar Arbune
Problem Statement 1: Vector Scalar Addition Iteration wise

```
#include <omp.h>
#include <stdio.h>
#include<stdlib.h>
int main() {
   int n=5;
   int a[n], c[n];
   int flag[n];
   printf("Total number of elements in the array is %d",n);

   for(int i=0;i<n;i++)
   {
        a[i]=i;
        flag[i]=0;
   }
        int scalar=10;
        omp_set_num_threads(5);
        #pragma omp parallel shared(c)
        for(int i = 0;i < n;i++) {
                if(!flag[i])
                {
                        c[i]=a[i]+scalar;
                        flag[i]=1;
                printf("\nThread number %d, executing iteration %d first
                        time",omp_get_thread_num(),i);

                }
                else
                {
                printf("\nThread number %d, executing iteration %d but  already
 done",omp_get_thread_num(),i);
                }

        }

        printf("\ni\ta[i]\t+scalar\t=\tb[i]\n");
     for(int i=0; i<n; i++)
        {
                printf("%d\t%d\t%d\t=\t%d\n", i, a[i],scalar,c[i]);
```

```
    }

}
```



Problem Statement 2: Shared Variable Vector Scalar Addition

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

int main()
{
        int n;
        printf("Enter how many number of elements in the array want : ");
        scanf("%d",&n);
        int a[n],b[n],c[n],i;

        for(i=0;i<n;i++)
        {
                a[i] = i;
                b[i] = i + 200;
        }
```

```
        #pragma omp parallel for shared(a,b,c) num_threads(n/3)
        for(i=0;i<n;i++)
        {
                c[i] = a[i] + b[i];
                printf("Thread %d works on element %d of the array\n",
omp_get_thread_num(), i);
        }

        printf("i\ta[i]\t+\tb\t=\tc[i]\n",b);
           for(i=0; i<n; i++) {
                printf("%d\t%d\t\t%d\t\t%d\n", i, a[i], b[i], c[i]);
   }
        return 0;
}
```

```
CG-DTE@CG-DTE-Student ~
$ g++ vector_shared.c -fopenmp -o vector_shared

CG-DTE@CG-DTE-Student ~
$ ./vector_shared.exe
Enter how many number of elements in the array want : 10
Thread 0 works on element 0 of the array
Thread 0 works on element 1 of the array
Thread 0 works on element 2 of the array
Thread 0 works on element 3 of the array
Thread 2 works on element 7 of the array
Thread 2 works on element 8 of the array
Thread 2 works on element 9 of the array
Thread 1 works on element 4 of the array
Thread 1 works on element 5 of the array
Thread 1 works on element 6 of the array
i       a[i]    +       b       =       c[i]
0       0               200             200
1       1               201             202
2       2               202             204
3       3               203             206
4       4               204             208
5       5               205             210
6       6               206             212
7       7               207             214
8       8               208             216
9       9               209             218

CG-DTE@CG-DTE-Student ~
$
```

Problem Statement 3: Shared Variable Vector Scalar Addition

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

int main()
{
        int n;
        printf("Enter how many numbers you want in array: ");
        scanf("%d",&n);
```

```
        int a[n],b,c[n],i;

        for(i=0;i<n;i++)
        {
                a[i] = i;
        }
        printf("\nEnter scalar: ");
        scanf("%d",&b);



        #pragma omp parallel for shared(a,b,c) num_threads(n/3)
        for(i=0;i<n;i++)
        {
                c[i] = a[i] + b;
                printf("Thread %d works on element %d of the array\n",
omp_get_thread_num(), i);
        }

        printf("i\ta[i]\t+\t%d\t=\tc[i]\n",b);
   for(i=0; i<n; i++) {
                printf("%d\t%d\t\t%d\t\t%d\n", i, a[i], b, c[i]);
   }
        return 0;
}
```

```
CG-DTE@CG-DTE-Student ~
$ g++ vector_scalar_shared.c -fopenmp -o vector_scalar_shared

CG-DTE@CG-DTE-Student ~
$ ./vector_scalar_shared.exe
Enter how many numbers you want in array: 10

Enter scalar: 20
Thread 0 works on element 0 of the array
Thread 0 works on element 1 of the array
Thread 0 works on element 2 of the array
Thread 0 works on element 3 of the array
Thread 2 works on element 7 of the array
Thread 2 works on element 8 of the array
Thread 2 works on element 9 of the array
Thread 1 works on element 4 of the array
Thread 1 works on element 5 of the array
Thread 1 works on element 6 of the array
i       a[i]    +       20      =       c[i]
0       0               20              20
1       1               20              21
2       2               20              22
3       3               20              23
4       4               20              24
5       5               20              25
6       6               20              26
7       7               20              27
8       8               20              28
9       9               20              29

CG-DTE@CG-DTE-Student ~
$
```

Problem Statement 4: Vector Scalar Addition by work sharing constraint

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

void input(int a[], int n,int no)
{
	cout<<"========Enter array "<<no<<"========\n";
	for(int i=0;i<n;i++)
	{
		cout<<i<<" element: ";
		cin>>a[i];
		cout<<endl;
	}
}

void display(int a[],int b[],int c[],int n)
{
	cout<<"========Addition========\na[i]\tb[i]\t=\tc[i]\n";
	for(int i=0;i<n;i++)
	{
		cout<<a[i]<<"\t"<<b[i]<<"\t=\t"<<c[i];
		cout<<endl;
	}
}

int main()
{
	int n;
	cout<<"Enter no of eleemnts in array: ";
	cin>>n;
	int a[n],b[n],c[n],i;

	for(i=0;i<n;i++)
	{
		a[i] = i;
		b[i] = i + 100;
	}

	#pragma omp parallel for firstprivate(a,b) shared(c) num_threads(n/2)
	for(i=0;i<n;i++)
	{
		c[i] = a[i] + b[i];
		printf("Thread %d works on element %d f the array\n",
omp_get_thread_num(), i);
	}

	display(a,b,c,n);
	return 0;
```

}

```
CG-DTE@CG-DTE-Student ~
$ g++ vector_vector_work_sharing.c -fopenmp -o vectorvector_worksharing

CG-DTE@CG-DTE-Student ~
$ ./vectorvector_worksharing.exe
Enter no of eleemnts in array: 10
Thread 1 works on element 2 of the array
Thread 1 works on element 3 of the array
Thread 2 works on element 4 of the array
Thread 2 works on element 5 of the array
Thread 4 works on element 8 of the array
Thread 4 works on element 9 of the array
Thread 3 works on element 6 of the array
Thread 3 works on element 7 of the array
Thread 0 works on element 0 of the array
Thread 0 works on element 1 of the array
========Addition========
a[i]    b[i]    =       c[i]
0       100     =       100
1       101     =       102
2       102     =       104
3       103     =       106
4       104     =       108
5       105     =       110
6       106     =       112
7       107     =       114
8       108     =       116
9       109     =       118

CG-DTE@CG-DTE-Student ~
$
```

Problem Statement 5: Vector Scalar addition FirstPrivate variable

```
CG-DTE@CG-DTE-Student ~
$ g++ work_sharing.c -fopenmp -o work_sharing

CG-DTE@CG-DTE-Student ~
$ ./work_sharing
Enter no of eleemnts in the array: 10

Enter scalar: 20
Thread 0 works on element 0 of the array
Thread 0 works on element 1 of the array
Thread 0 works on element 2 of the array
Thread 0 works on element 3 of the array
Thread 2 works on element 7 of the array
Thread 1 works on element 4 of the array
Thread 1 works on element 5 of the array
Thread 1 works on element 6 of the array
Thread 2 works on element 8 of the array
Thread 2 works on element 9 of the array
i       a[i]    +       20      =       c[i]
0       0               20              20
1       1               20              21
2       2               20              22
3       3               20              23
4       4               20              24
5       5               20              25
6       6               20              26
7       7               20              27
8       8               20              28
9       9               20              29

CG-DTE@CG-DTE-Student ~
```

Data Scooping:
1. Private (Variable List) : Specify variable local to each Thread.
2. Firstprivate(Variable List) : Similar to private variable , private variable are initialized to variable before parallel directives.
3. Shared(Variable List): Specify variable that are shared among all threads.

SPMP(Single Program Multiple Data):
Each thread executes own data.

Work sharing :
Is to split up pathways through the code between thread within a thread.

Github Link:https://github.com/shwetaarbune/HPC-LAB