Experiment 3

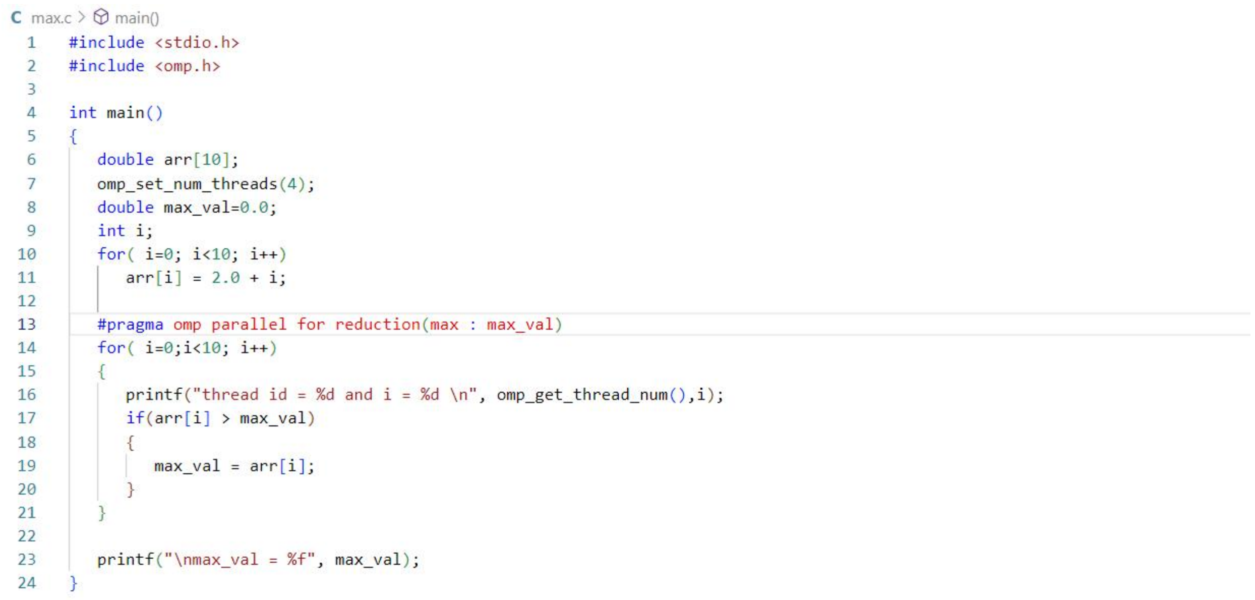
Implement Min, Max, Sum and Average operations using Parallel Reduction

1. max.c

```
#include <stdio.h>
#include <omp.h>
int main()
{
  double arr[10];
  omp_set_num_threads(4);
  double max_val=0.0;
  int i;
  for( i=0; i<10; i++)
    arr[i] = 2.0 + i;
  #pragma omp parallel for reduction(max : max_val)
  for( i=0;i<10; i++)
  {
    printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
    if(arr[i] > max_val)
    {
      max_val = arr[i];
    }
  }
  printf("\nmax_val = %f", max_val);
}
```

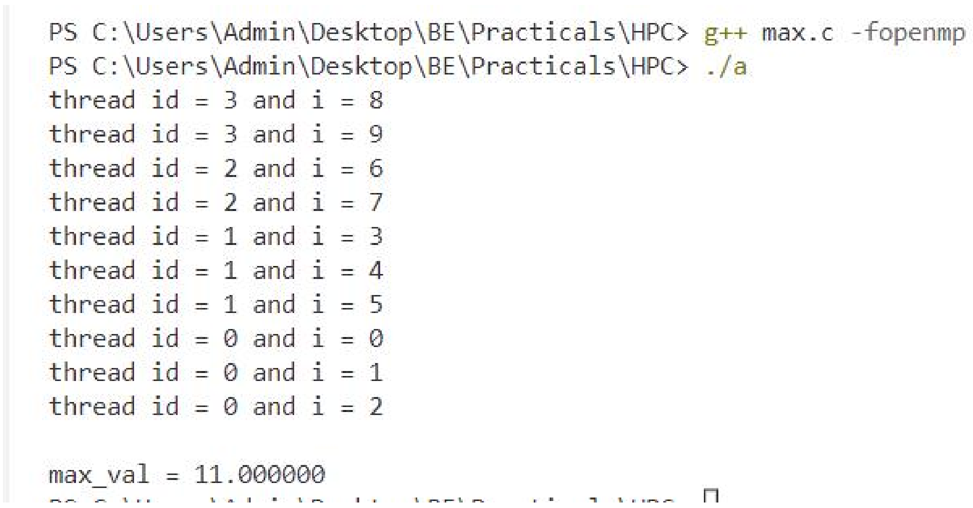
Output:

```
user1@user1-ThinkCentre-E73:~$ g++ max.c -fopenmp
user1@user1-ThinkCentre-E73:~$ ./a.out
thread id = 3 and i = 8
thread id = 3 and i = 9
thread id = 0 and i = 0
thread id = 0 and i = 1
thread id = 0 and i = 2
thread id = 1 and i = 3
thread id = 1 and i = 4
thread id = 1 and i = 5
thread id = 2 and i = 6
thread id = 2 and i = 7
max_val = 11.000000
```



max.c > main()

```c
#include <stdio.h>
#include <omp.h>

int main()
{
    double arr[10];
    omp_set_num_threads(4);
    double max_val=0.0;
    int i;
    for( i=0; i<10; i++)
        arr[i] = 2.0 + i;

    #pragma omp parallel for reduction(max : max_val)
    for( i=0;i<10; i++)
    {
        printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
        if(arr[i] > max_val)
        {
            max_val = arr[i];
        }
    }

    printf("\nmax_val = %f", max_val);
}
```



```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ max.c -fopenmp
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
thread id = 3 and i = 8
thread id = 3 and i = 9
thread id = 2 and i = 6
thread id = 2 and i = 7
thread id = 1 and i = 3
thread id = 1 and i = 4
thread id = 1 and i = 5
thread id = 0 and i = 0
thread id = 0 and i = 1
thread id = 0 and i = 2

max_val = 11.000000
```

2. min.c

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

int main()
{
  double arr[10];
  omp_set_num_threads(4);
  double min_val=9.0;
  int i;

  for( i=0; i<10; i++)
     arr[i] = 2.0 + i;

  #pragma omp parallel for reduction(min : min_val)
  for( i=0;i<10; i++)
  {
     printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
     if(arr[i] < min_val)
     {
        min_val = arr[i];
     }
  }
  printf("\nmin_val = %f", min_val);
}
```

Output:

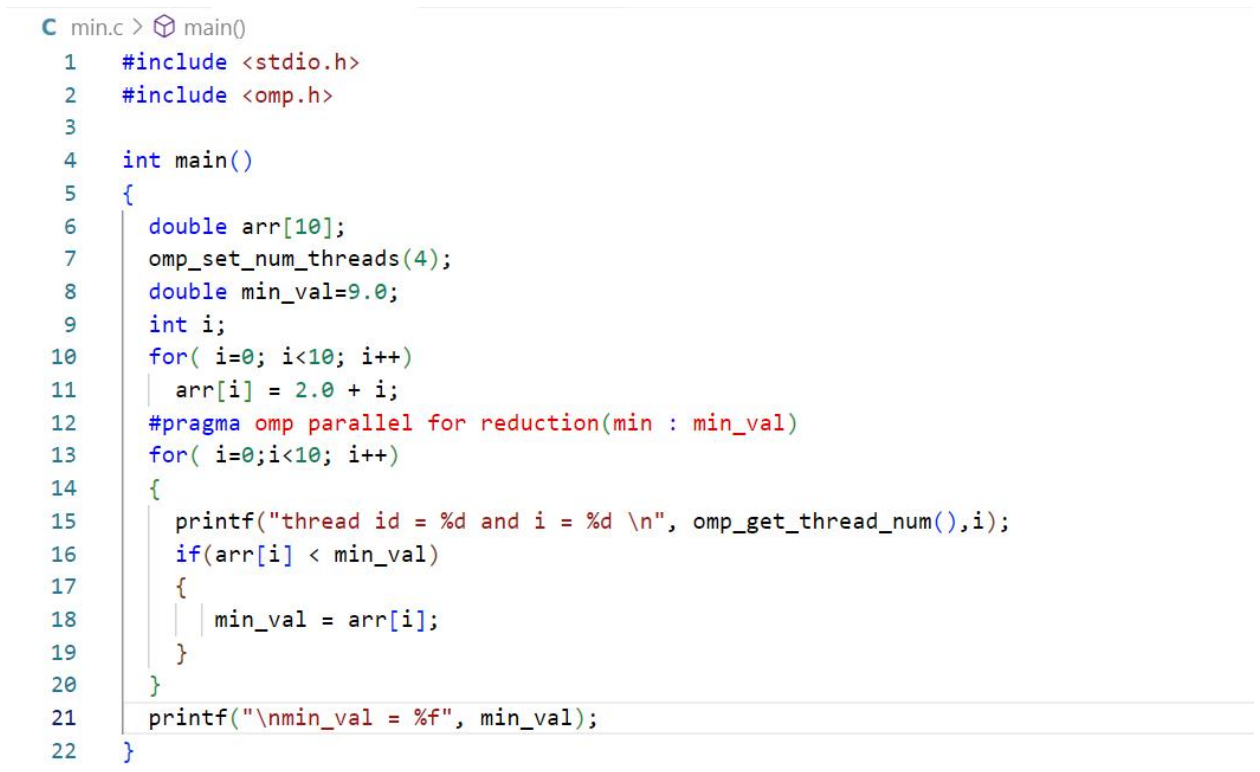
```
user1@user1-ThinkCentre-E73:~$ g++ min.c -fopenmp
user1@user1-ThinkCentre-E73:~$ ./a.out
thread id = 2 and i = 6
thread id = 2 and i = 7
thread id = 0 and i = 0
thread id = 0 and i = 1
thread id = 0 and i = 2
thread id = 3 and i = 8
```

thread id = 3 and i = 9

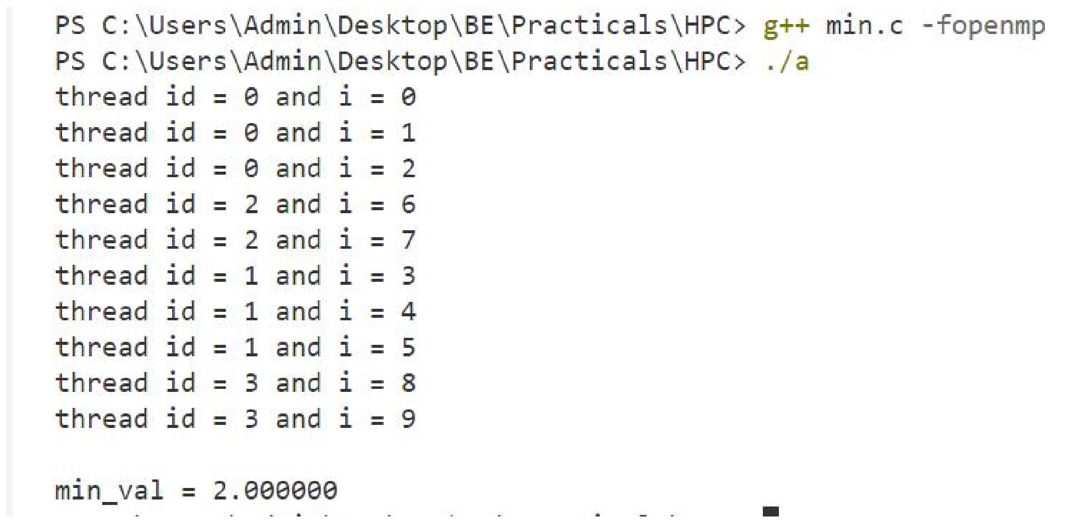
thread id = 1 and i = 3

thread id = 1 and i = 4

thread id = 1 and i = 5



```
min.c > main()
#include <stdio.h>
#include <omp.h>

int main()
{
    double arr[10];
    omp_set_num_threads(4);
    double min_val=9.0;
    int i;
    for( i=0; i<10; i++)
      arr[i] = 2.0 + i;
    #pragma omp parallel for reduction(min : min_val)
    for( i=0;i<10; i++)
    {
      printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
      if(arr[i] < min_val)
      {
         min_val = arr[i];
      }
    }
    printf("\nmin_val = %f", min_val);
}
```



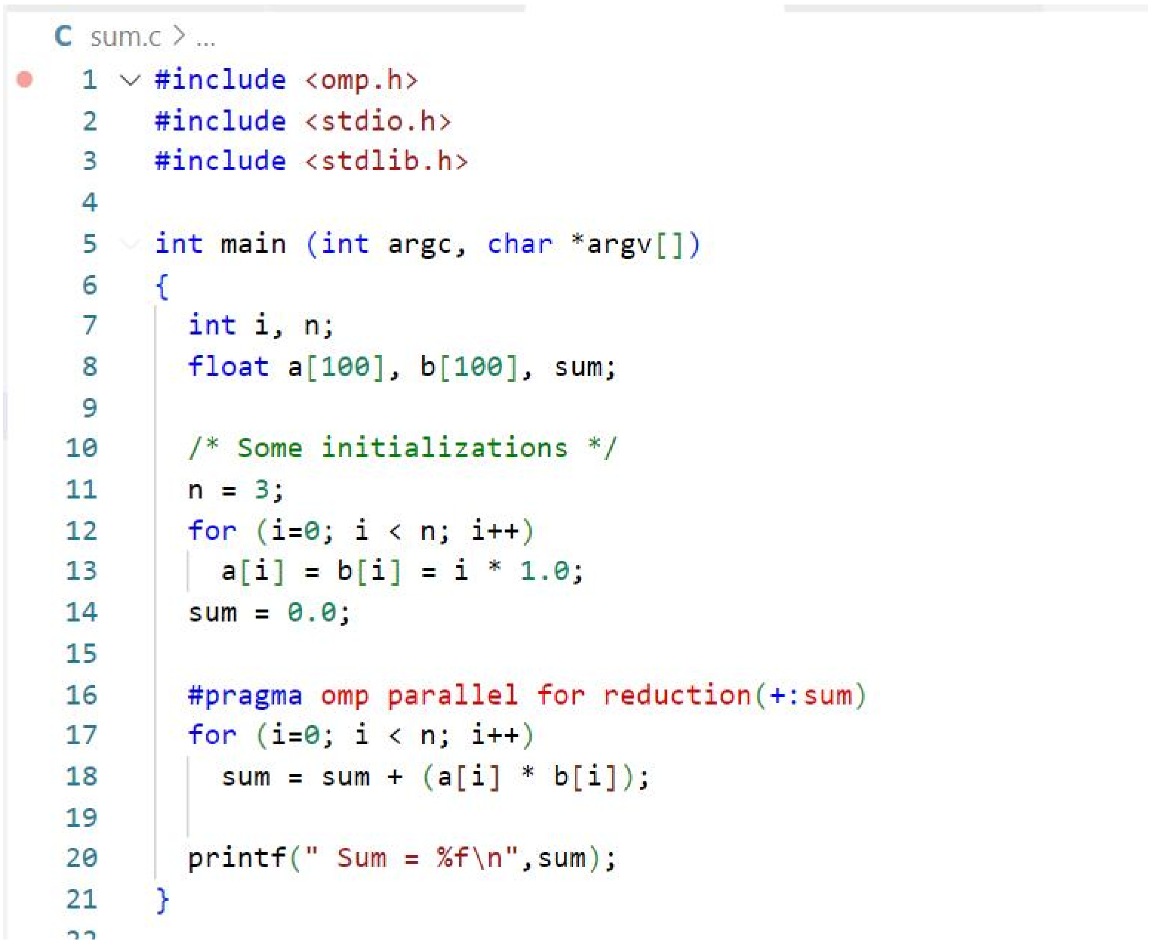
```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ min.c -fopenmp
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
thread id = 0 and i = 0
thread id = 0 and i = 1
thread id = 0 and i = 2
thread id = 2 and i = 6
thread id = 2 and i = 7
thread id = 1 and i = 3
thread id = 1 and i = 4
thread id = 1 and i = 5
thread id = 3 and i = 8
thread id = 3 and i = 9

min_val = 2.000000
```

3. sum.c

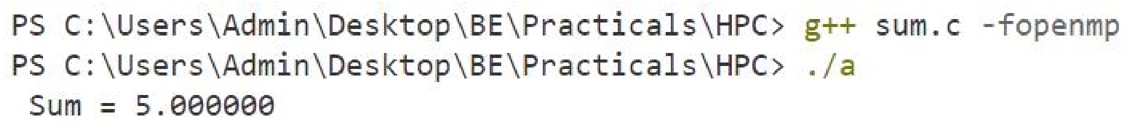
```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
  int i, n;
  float a[100], b[100], sum;

  /* Some initializations */
  n = 3;
  for (i=0; i < n; i++)
    a[i] = b[i] = i * 1.0;
  sum = 0.0;

  #pragma omp parallel for reduction(+:sum)
  for (i=0; i < n; i++)
    sum = sum + (a[i] * b[i]);

  printf(" Sum = %f\n",sum);
}
```

Output:

```
user1@user1-ThinkCentre-E73:~$ g++ sum.c -fopenmp
user1@user1-ThinkCentre-E73:~$ ./a.out
 Sum = 5.000000
```


sum.c > ...

```c
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
  int i, n;
  float a[100], b[100], sum;

  /* Some initializations */
  n = 3;
  for (i=0; i < n; i++)
    a[i] = b[i] = i * 1.0;
  sum = 0.0;

  #pragma omp parallel for reduction(+:sum)
  for (i=0; i < n; i++)
    sum = sum + (a[i] * b[i]);

  printf(" Sum = %f\n",sum);
}
```



```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ sum.c -fopenmp
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
 Sum = 5.000000
```

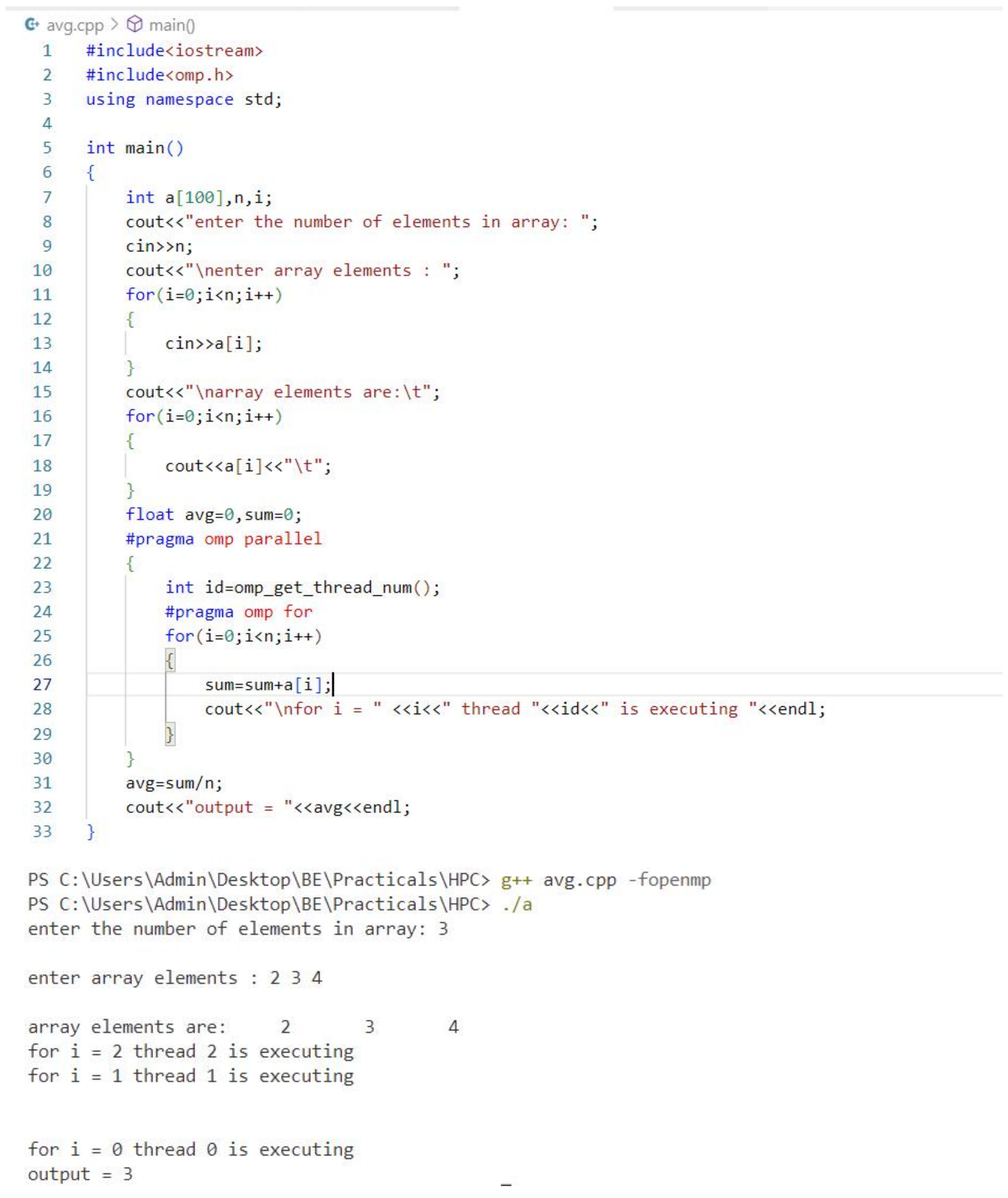
4. avg.cpp

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

int main()
{
   int a[100],n,i;
   cout<<"enter the number of elements in array: ";
   cin>>n;
   cout<<"\nenter array elements : ";
   for(i=0;i<n;i++)
   {
      cin>>a[i];
   }
   cout<<"\narray elements are:\t";
   for(i=0;i<n;i++)
   {
      cout<<a[i]<<"\t";
   }
   float avg=0,sum=0;
   #pragma omp parallel
   {
      int id=omp_get_thread_num();
      #pragma omp for
      for(i=0;i<n;i++)
      {
         sum=sum+a[i];
         cout<<"\nfor i = " <<i<<" thread "<<id<<" is executing "<<endl;
      }
   }
   avg=sum/n;
   cout<<"output = "<<avg<<endl;
}
```

Output:

```
enter the number of elements in array : 5
```

```
 enter array elements : 3 4 6 7 8
 array elements are: 3 4          6          7          8
for i= 0 thread 0 is executing
for i= 2 thread 1 is executing
for i= 3 thread 2 is executing
for i= 4 thread 3 is executing
for i= 1 thread 0 is executing
output =  3.4
```



avg.cpp > main()

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

int main()
{
    int a[100],n,i;
    cout<<"enter the number of elements in array: ";
    cin>>n;
    cout<<"\nenter array elements : ";
    for(i=0;i<n;i++)
    {
        cin>>a[i];
    }
    cout<<"\narray elements are:\t";
    for(i=0;i<n;i++)
    {
        cout<<a[i]<<"\t";
    }
    float avg=0,sum=0;
    #pragma omp parallel
    {
        int id=omp_get_thread_num();
        #pragma omp for
        for(i=0;i<n;i++)
        {
            sum=sum+a[i];
            cout<<"\nfor i = " <<i<<" thread "<<id<<" is executing "<<endl;
        }
    }
    avg=sum/n;
    cout<<"output = "<<avg<<endl;
}
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ avg.cpp -fopenmp
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
enter the number of elements in array: 3

enter array elements : 2 3 4

array elements are:     2       3       4
for i = 2 thread 2 is executing
for i = 1 thread 1 is executing


for i = 0 thread 0 is executing
output = 3
```