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

void vector_add(int* a, int* b, int* c)
{
   #pragma omp parallel for
   for(int i =0; i < 10; i++)
   {
      c[i] = a[i] + b[i];
      printf("a%d + b%d = \n", i, i);
      printf("%d + %d = %d\n", a[i], b[i], c[i]);
   }

}

int main()
{
   int a[10];
   int b[10];
   for(int i = 0; i < 10; i++)
   {
      a[i] = rand()%20;
      b[i] = rand()%20;
   }
   int* c = (int*)malloc(10* sizeof(int));
   vector_add(a, b, c);
   for(int i =0; i < 10; i++)
   {
      printf("c[%d] = %d", i, c[i]);
   }

}
```

```
> gcc vector.c -fopenmp -o vector
> ./vector
a0 + b0 =
3 + 6 = 9
a1 + b1 =
17 + 15 = 32
a2 + b2 =
13 + 15 = 28
a6 + b6 =
10 + 19 = 29
a5 + b5 =
2 + 7 = 9
a4 + b4 =
9 + 1 = 10
a3 + b3 =
6 + 12 = 18
a8 + b8 =
```

```
0 + 6 = 6
a7 + b7 =
3 + 6 = 9
a9 + b9 =
12 + 16 = 28
```

```
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>
#define NUM_THREADS 4
static long num_steps= 1000000; double step;

void main()
{
   int i; double x, pi, sum[NUM_THREADS];
   step = 1.0/(double) num_steps;
   #pragma omp parallel private(i,x)
   {
      int id = omp_get_thread_num();
      for(i=id, sum[id]=0.0; i < num_steps; i=i+NUM_THREADS)
      {
         x= (i+0.5)*step;
         sum[id] += 4.0/(1.0+x*x);
      }
   }
   for(i=1; i <NUM_THREADS; i++)
      sum[0] += sum[i];
      pi = sum[0] /num_steps;
      printf("pi = %1.12f\n", pi);

}
```

```
> gcc thread_pi.c -fopenmp -o thread_pi
anubhav@bracer:~/PP Lab> ./thread_pi
pi = 3.141592653590
```