**Simple Program:**

#include <stdio.h>

#include <omp.h>

int main(int argc, char\*\* argv){

#pragma omp parallel

{

printf(“Hello from process: %d\n”, omp\_get\_thread\_num());

}

return 0;

}

**//compute the sum of two arrays in parallel**

#include < stdio.h >

#include < omp.h >

#define N 10000

int main(void) {

float a[N], b[N], c[N];

int i;

/\* Initialize arrays a and b \*/

for (i = 0; i < N; i++) {

a[i] = i \* 2.0;

b[i] = i \* 3.0;

}

/\* Compute values of array c = a+b in parallel. \*/

#pragma omp parallel shared(a, b, c) private(i)

{

#pragma omp for

for (i = 0; i < N; i++) {

c[i] = a[i] + b[i];

printf ("%f\n", c[i]);

}

}

}

**Given a nxn matrix A and a vector x of length n, their product y=A⋅x. Write a program to implement the multiplication using OpenMP PARALLEL directive.**

#include<stdio.h>

#include<stdlib.h>

#include<omp.h>

void main() {

int m,n;

printf("Enter the size of square matrix : ");

scanf("%d",&n);

printf("Enter the size of vector : ");

scanf("%d", &m);

if (m!=n) {

printf("Multiplication is not possible.\n");

exit(0);

}

int i=0,j=0;

int \*\*arr=(int\*\*)malloc(n\*sizeof(int\*));

int \*vec=(int\*)malloc(n\*sizeof(int));

int \*res=(int\*)malloc(n\*sizeof(int));

omp\_set\_num\_threads(n);

#pragma omp parallel private(j)

{

#pragma omp for

for (i=0;i<n;i++) {

srand(i);

arr[i]=(int\*)malloc(n\*sizeof(int));

vec[i]=rand()%100;

for (j=0;j<n;j++)

arr[i][j]=rand()%100;

}

}

#pragma omp parallel private(j)

{

#pragma omp for

for(i=0;i<n;i++) {

res[i]=0;

for(j=0;j<n;j++)

res[i]+=arr[i][j]\*vec[j];

}

}

printf("Matrix \* Vector = Resultant Matrix\n");

for(i=0;i<n;i++) {

for(j=0;j<n;j++)

printf("%3d ",arr[i][j]);

if(i==n/2)

printf(" \* %3d = %6d\n",vec[i],res[i]);

else

printf(" %3d %6d\n",vec[i],res[i]);

}

}