**Name : Ibadullah Shaikh**

**Roll No : 19K-0259**

**Section : BCS-4H**

**Q1 (a) Addition of Matrix:**

**Code:**

#include<stdio.h>

#include<omp.h>

#define N 3

int main()

{

int mat1[N][N],mat2[N][N],addition[N][N];

printf("Enter Matrix 1\n");

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

{

for(int j=0;j<N;j++)

{

scanf("%d",&mat1[i][j]);

}

}

printf("Enter Matrix 2\n");

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

{

for(int j=0;j<N;j++)

{

scanf("%d",&mat2[i][j]);

}

}

printf("Matrix 1:\n");

for(int i=0;i<N\*N;i++)

{

if(i%3==0)

{

printf("\n");

}

printf("%d ",\*(\*mat1+i));

}

printf("\n\n");

printf("Matrix 2:\n");

for(int i=0;i<N\*N;i++)

{

if(i%3==0)

{

printf("\n");

}

printf("%d ",\*(\*mat2+i));

}

printf("\n\n");

#pragma omp parallel

#pragma omp for

for(int i=0;i<N\*N;i++)

{

\*(\*addition+i)=\*(\*mat1+i)+\*(\*mat2+i);

}

printf("Thrads: %d\n",omp\_get\_num\_threads());

for(int i=0;i<N\*N;i++)

{

if(i%3==0)

{

printf("\n");

}

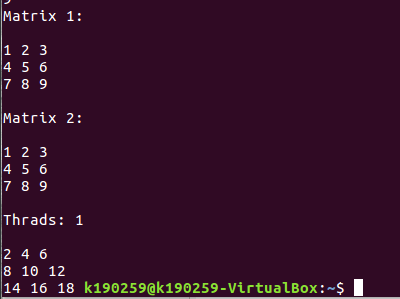
printf("%d ",\*(\*addition+i));

}

return 0;

}

**Screenshot:**



**Q1 (b) Multiplication of Matrix**

**Code:**

#include<stdio.h>

#include<omp.h>

#define N 3

int matrix1[N][N];

int matrix2[N][N];

int multi[N][N];

int multiply(int i , int j)

{

int sum=0;

for(int k=0;k<N;k++)

{

sum=sum+(matrix1[i][k]\*matrix2[k][j]);

}

return sum;

}

int main()

{

printf("Enter Matrix 1\n");

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

{

for(int j=0;j<N;j++)

{

scanf("%d",&matrix1[i][j]);

}

}

printf("Enter Matrix 2\n");

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

{

for(int j=0;j<N;j++)

{

scanf("%d",&matrix2[i][j]);

}

}

printf("Matrix 1:\n");

for(int i=0;i<N\*N;i++)

{

if(i%3==0)

{

printf("\n");

}

printf("%d ",\*(\*matrix1+i));

}

printf("\n\n");

printf("Matrix 2:\n");

for(int i=0;i<N\*N;i++)

{

if(i%3==0)

{

printf("\n");

}

printf("%d ",\*(\*matrix2+i));

}

printf("\n\n");

#pragma omp parallel

#pragma omp for

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

{

for(int j=0;j<N;j++)

{

multi[i][j]=multiply(i,j);

}

}

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

{

for(int j=0;j<N;j++)

{

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

}

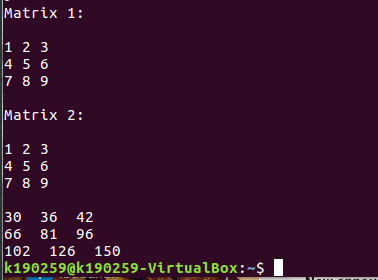
printf("\n");

}

return 0;

}

**Screenshot:**



**Q2: Taylor into OpenMp**

**Code:**

#include<stdio.h>

#include<omp.h>

#include<math.h>

#include<stdlib.h>

int i,term;

long double factorial[100],power[100];

long double final[100];

long double FACTORIAL(long double temp)

{

if(temp <= 1)

{

return 1;

}

return temp\*FACTORIAL(temp-1);

}

long double power1(double num,int power)

{

long double res= 1;

for(int i=0;i<power;i++)

{

res=res\*num;

}

return res;

}

long double sum=0;

int main()

{

double x = 4;

#pragma omp parallel

#pragma omp for

for(int i=0;i<15;i++)

{

factorial[i]=FACTORIAL(i);

power[i]=i;

final[i]= power1(x,(int)power[i])/factorial[i] ;

printf("Term %d %Lf\n",i+1,final[i]);

sum=sum+final[i];

}

printf("Total value of e^%f: %Lf\n",x,sum);

return 0;

} **Screenshot:**

