Tarea Programada 3.
Programación Paralela y Concurrente
Dayana Marín Mayorga
B64096

B	C	D	E	F	G	H	I
Términos	Versiones				Speedup		
N	**Serial**	**OpenMP**	**Pthreads**	**MPI**	**OpenMP**	**Pthreads**	**MPI**
2	0,38354396	0,28436931	0,10615478	0,33766156	1,34875302	3,6130634	1,13588281
4	0,61151131	0,40600445	0,44101596	0,40540997	1,50616897	1,38659678	1,50837759
16	0,66379468	0,51590888	0,55306538	0,61449538	1,28665101	1,20021016	1,0802273
32	0,77247875	0,8055254	0,61178795	0,72927766	0,95897503	1,26265767	1,0592382
64	0,9917311	0,83855156	0,79775341	1,28942112	1,18267158	1,24315494	0,76912894

Observaciones: Los mejores tiempos se dieron con la versión Pthreads, pero en ciertas ocasiones con OpenMP se obtuvieron resultados similares o mejores.

Con Respecto al SpeedUp también se obtuvieron mejores resultados con la versión pthreads en cuanto a velocidad y en algunas con OpenMP.

Por limitantes en cuanto a los recursos de la computadora se hizo un cambio en la escala logarítmica que propuso el profesor en el enunciado, por potencias de dos saltando el 8.

2. Lógica de la tarea:
Para cada versión se realiza cambios en el main, pero siempre se utiliza la clase Numeros.cc que cuenta con los siguientes métodos.

Método uno:
Este método se encarga de averiguar si un número es primo, tiene un contador que verifica que si un número es divisible más de dos veces entonces no es primo, sino si es primo.

```
int Numeros::esPrimo(int numero){
int primo;
int contador = 0;
for(int i = 1; i <= numero; i++){
      if(numero % i == 0){
      contador++;
}
}
if (contador > 2){
primo = 0;
}else{
primo = 1;
}
return primo;
}
```

Método dos:
En este método lo que se hace son 3 for anidados, el primer for representa los números pares y los dos últimos for representan los números primos.

```
vector<vector<int>> Numeros::calcular(int limiteSuperior){
vector<vector<int>> sumas;
vector<int> suma;
//for para los pares
for (int i = 6; i <= limiteSuperior; i++){
        if(i%2 == 0){// se verifica que es par
           // for para ver los primos y la suma;
            for(int j = 2; j <= limiteSuperior; j++){
              for(int k = 2; k<= limiteSuperior; k++){
                   if(esPrimo(k) == 1 && esPrimo(j)){// Se accede al metodo esprimo para ver si es primo.
                       if(j+k == i){//Se verifica si la suma es igual al numero par que estamos verificando
                          //printf( "Valor: %d = %d + %d \n",i,j,k);
                          suma.push_back(i);
                          suma.push_back(j);
                          suma.push_back(k);
                          //suma es un vector que guarda el numero par y los dos numeros primos que forman la suma
                          sumas.push_back(suma);
                          suma.clear();
}
}
}
}
}
}
return sumas;
}
```

Para las versiones de OpenMP, MPI y Pthreads se cambió el método calcular de la siguiente manera:

Se le agrega un nuevo parámetro que indica el inicio y el fin de rango con los cuales se van hacer los cálculos.

```
vector<vector<int>> Numeros::calcular(int inicio,int limiteSuperior){
vector<vector<int>> sumas;
vector<int> suma;
for (int i = inicio; i <= limiteSuperior; i++){
        if(i%2 == 0){
            for(int j = 2; j <= limiteSuperior; j++){
              for(int k = 2; k<= limiteSuperior; k++){
                   if(esPrimo(k) == 1 && esPrimo(j)){
                       if(j+k == i){
                          //printf( "Valor: %d = %d + %d \n",i,j,k);
                          suma.push_back(i);
                          suma.push_back(j);
                          suma.push_back(k);
                          sumas.push_back(suma);
                          suma.clear();
}
}
}
}
}
}
return sumas;
}
```