não usar	
1	
2	
3	
4	
T	

1. Discuta a veracidade da afirmação abaixo, justificando a sua resposta:
   "O ganho de desempenho obtido com a vectorização de código, relativamente à respectiva versão escalar, deve-se apenas à diminuição do número médio de ciclos por instrução (CPI)."

Nome: _____ Número:_____

2. Para cada um dos ciclos abaixo indique justificando se é vectorizável. Se identificar dependências de dados entre iterações calcule a respectiva distância e indique o seu tipo (*Write After Read* (WAR) ou *Read After Write* (RAW)).

```
struct {int g, h; } a[SIZE];

for (i=0 ; i <SIZE-1 ; i++)
  a[i].g = 20 * a[i+1].h;
```

```
float a[SIZE];

for (i=SIZE-1 ; i >= 2 ; i--)
  a[i] = 5 * a[i] / a[i-2];
```

3. Considere código C + OpenMP apresentado abaixo. Note que:

- a cláusula `#pragma omp single` garante que o bloco que se segue é executado apenas por **uma *thread***, podendo esta ser qualquer *thread* do *team*; note ainda que **esta cláusula implica uma barreira no fim**, isto é as *threads* só prosseguem para as instruções seguintes quando todas as *threads* atingirem o fim deste bloco.

```
#pragma omp parallel
{  int i, first=false, tid = omp_get_thread_num ();
   double T;
   printf ("Thread %d starting\n", tid);
   #pragma omp for
     for (i=0; i < 300000 ; i++) do_work(i);
   #pragma omp single
   {  T = omp_get_time ();
     first = true;
     printf ("Thread %d work done\n", tid);
   }
   if (first) printf ("1st finished %.0lf us before\n", (omp_get_wtime()-T)*1e6);
   printf ("Thread %d finishing\n", tid);
}
```

Considere as 3 opções de *output* apresentadas abaixo, para uma execução com 3 *threads*. Apenas uma destas opções é possível. Indique, **justificando**, qual.

OPÇÃO A	**OPÇÃO B**	**OPÇÃO C**
Thread 1 starting	Thread 1 starting	Thread 1 starting
Thread 0 starting	Thread 0 starting	Thread 0 starting
Thread 0 work done	Thread 2 starting	Thread 2 starting
1st finished 7 us before	Thread 0 work done	Thread 2 finishing
Thread 2 starting	Thread 2 finishing	Thread 0 work done
Thread 2 finishing	1st finished 7 us before	1st finished 7 us before
Thread 1 finishing	Thread 1 finishing	Thread 1 finishing
Thread 0 finishing	Thread 0 finishing	Thread 0 finishing

Nome: _____ Número:_____

4. Para o código abaixo proponha uma implementação que explore *Thread Level Parallelism* usando o OpenMP. Justifique as suas opções.

```
#define S 1000000
float a[S];
int i ,j;
for (i=1 ; i <S ; i++) {
  a[i] = 0.;
  for (j=0 ; j < i ; j++) {
    a[i] += j;
} }
```