Homework1 (Due Wednesday, May 30):

1. (20 points) Write a parallel program to sum n numbers in an array using n processors.

```
for (i = 0; i < log(n); i++)
{
  for j % 2 ^ (i + 1) == 0 do in parallel
  {
     Sum[j] = Sum[j] + Sum[j + 2 ^ i];
  }
}
```

T = O(log(n))

2. (20 points) Write a parallel program to sum n numbers in an array using n/32 processors.
   (1) Group by 32 numbers in each group and there are n/32 groups with n/32 processors. Sequentially sum 32 numbers need 31 operations.

```
# do in parallel for 32 groups
for (i = 0 to n/32;) do in parallel
{
    # sequentially sum 32 numbers in each group
    for(j = 0; j < 32; j++)
    {
        Sum[i] = Sum[i] + Num[j];
    }
}
```

   T = O(32) (in fact 31)

   (2) Get sum of each group and there are n/32 numbers left with n/32 processors. According to last question:

```
for (i = 0; i < log(n/32); i++)
{
  for j % 2 ^ (j + 1) == 0 do in parallel
  {
     Sum[j] = Sum[j] + Sum[j + 2 ^ i];
  }
}
```

T = O(log(n/32)) <= O(log(n))

3. (20 points) Write a parallel program to sum n numbers in an array using p <n processors.
   (1) Group by n/p numbers in each group and there are p groups with p processors. Therefore, each group will have 1 processor. Sequentially sum n/p numbers need n/p-1 operations.

```
 # do in parallel for p groups
for (i = 0 to p;) do in parallel
 {
     # sequentially sum n/p numbers in each group
    for(j = 0; j < n/p; j++)
    {
        Sum[i] = Sum[i] + Num[j];
    }
}
```

   T = O(n/p) (in fact n/p-1)

   (2) Get sum of each group and there are p numbers left with p processors. According to last question:

```
 for (i = 0; i < log(p); i++)
 {
   for j % 2 ^ (i + 1) == 0 do in parallel
   {
      Sum[j] = Sum[j] + Sum[j + 2 ^ i];
   }
}
```

   T = O(log(p))

T = n/p + log(p) <= O(n)

4. (20 points) Write a parallel program to find the minimum of n numbers in an array using p<=n processors.
   (1) Group by n/p numbers in each group and there are p groups with p processors. Therefore, each group will have 1 processor. Sequentially compare n/p numbers need n/p - 1 operations to get the minimum value.

```
 # do in parallel for p groups
initialize min[p]
for (i = 0 to p;) do in parallel
```

```
{
    # sequentially compare n/p numbers in each group
    for(j = 0; j < n/p; j++)
    {
        if(min[i] > Num[j])
        {
            min[i] = Num[j];
        }
    }
}
```

T = O(n/p) (in fact n/p-1)

(2) Compare of each group's min and there are p numbers left with p processors. Use recursion, group n^((1/2)^i):

```
Find(1);

Find(i, number(i)){
    if(n^((1/2)^i) != 1)
    {
        Number(i) = Find(i+1, number(i+1));
        return Number(i);
    }
    else{
        return min;
    }
}
```

T = O(log(log(p))

T = n/p + log(log(p)) <= O(n)

5. (20 points) Describe a parallel algorithm to find the minimum of n numbers in an array using p > n processors on a concurrent write parallel computation model.

Divided into T groups and each group have n/T numbers. (n/T)^2*T = p. Therefore, T=n^2/p. Then divided into T groups of n^2/p numbers, (n^2/(T*P))^2*T=p. Therefore,T = n^4/p^3…Iteratively， i = n^(2^i)/p^(2^(i-1)) = 1 and i = log(logp/log(p/n))

```
 Divide n numbers into T groups;
Calculate number of groups as (n/T)^2*T = p
for(i=1;i<log(logp/log(p/n));i++)
{
   T=N^(2^i)/p^(2^(i-1));
```

```
    Divide numbers left (same to the number of last group) into T groups
    Find min of each group for next loop with constant time;
}
```