HPC Home Assignment Exercise 1

Yuxuan Jing 1454901

1.1

```
base = n / p;
remainder = n % p;
for (any core){
    nthread = get_thread_num();
    if (nthread < reminder){
        my_first_i = nthread * (base + 1);
        my_last_i = (nthread + 1) * (base + 1);
    }
    else{
        my_first_i = nthread * base + remainder;
        my_last_i = (nthread + 1) * base + remainder;
    }
}
```

1.2

```
//solution 1
n_tot = (1+n)*n/2;
each_core_work = n_tot/p;
my_first_i[p];
my_last_i[p];
sum = 0
my_first_i.append(0)
for (size_t i = 0; i < p; i++){
    sum += i + 1;
    if (sum > each_core_work){
        my_last_i.append(i-1)
    }
    sum = i + 1;
    my_first_i.append(i)
}
my_last_i.append(p-1);

for(each_core){
    nthread = get_thread_num();
    my_first_i = my_first_i[nthread]
    my_last_i = my_last_i[nthread]
}

//solution 2

n_each = n/p + (n%p != 0);
assignment[p][n_each];
for (size_t i = 0; i < p; i++){
    if (i%(2*p) < p){
        assignment[i%p].append(i);
    }
    else{
        assignment[p - i%p - 1].append(i);
    }
}

//solution 3

n_each = n/p + (n%p != 0);
assignment[p][n_each];
for (size_t i = 0; i < p; i++){
    if (i%(2*p) < p){
        assignment[i%p].append(n-i);
    }
    else{
        assignment[p - i%p - 1].append(n-i);
    }
}
```

1.3

```
int divisor = 1;
int n_thread = get_thread_total();
while (divisor < n_thread){
    for (size_t i = 0; i < n_thread, i += 2 * divisor){
        if (i % (2*divisor) == 0){
            sum(core i) += sum(core i + divisor);
        }
        // else{
        //     send sum to core i - divisor;
        // }
    }
    divisor *= 2;
}
```

1.4

```
bit = 0;
int n_thread = get_thread_total();

while ( (1 << bit) < n_thread){
    bitmask = 1 << bit;
    for (size_t i = 0; i < n_thread, i += bitmask){
        if ( (i & bitmask) == 0){
            sum(core i) += sum(core i + bitmask);
        }
        // else{
        //     send sum to core i - bitmask;
        // }
    }
    bit += 1;
}
```

1.5

```
int divisor = 1;
int n_thread = get_thread_total();
while (divisor < n_thread){
    for (size_t i = 0; i < n_thread, i += 2 * divisor){
        if (i % (2*divisor) == 0){
            if (i + divisor < n_thread){  // adding for this problem
                sum(core i) += sum(core i + divisor);
            }
        }
    }
    divisor *= 2;
}


bit = 0;
int n_thread = get_thread_total();
while ( (1 << bit) < n_thread){
    bitmask = 1 << bit;
    for (size_t i = 0; i < n_thread, i += bitmask){
        if ( (i & bitmask) == 0){
            if (i + bitmask < n_thread){  // adding for this problem
                sum(core i) += sum(core i + bitmask);
            }

        }
    }
    bit += 1;
}
```

1.6

```
p = n_thread_total();
n1 = p - 1;
n2 = log2(p) + !((double)(int)log2() == log());

p       Original    Tree
2^1     2^1-1       1
2^2     2^2-1       2
2^3     2^3-1       3
2^4     2^4-1       4
2^5     2^5-1       5
2^6     2^6-1       6
2^7     2^7-1       7
2^8     2^8-1       8
2^9     2^9-1       9
2^10    2^10-1      10
```

1.7

```
combination of task- and data- parallelism.
task-parallelism: send sum and receive sum, add sum at the same time.
data-parallelism: receive nodes are doing the same kinds of task: adding two sums.
```

1.8

```
a.
cleaning the house
set the room
prepare food

b.
5 people clean 5 parts of the department at the same time.

c.
5 people abcde
clean 5 parts in the department
then
ab prepare the food while cde set the rooms

        a       b       c       d       e
time1   clean1  clean2  clean3  clean4  clean5
time2   food1   food2   set1    set2    set3
time3   food1   food2   set1    set2    set3
time4   food1   food2   set1    set2    set3
```