Lab5 – openMP

e.g. 1 sorting – parallelize with OpenMP.

```
#define N 8

void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

main()
{
    int A[N], stage, i;

    for (i=0; i < N; i++)
        A[i] = N-i;

    for (stage = 0; stage < N; stage++)
        if (stage % 2 == 1)
            for (i = 1; i < N; i += 2)
                if (A[i-1] > A[i])
                    swap(&A[i], &A[i-1]);
        else
            for (i = 1; i < N-1; i += 2)
                if (A[i] > A[i+1])
                    swap(&A[i], &A[i+1]);

    for (i=0; i < N; i++) printf("%d ", A[i]); printf("₩n");
}
```

e.g. 2 confliction - parallelize with OpenMP properly.

```
#define N 13
int printf(char *, ...);
int f(int i)
{
   i = i*2;
   return i;
}

main()
{
    int A[N], B[N], i;
    omp_set_num_threads(4)
;
    A[0] = f(0);
    B[0] = f((0);
    #pragma omp parallel for
    for (i=1; i<N; i++) {
        A[i] = f(i);
        B[i] = A[i-1] + A[i];
    }

    for (i=0; i<N; i++)
        printf("%d: %d %d₩n", i, A[i], B[i]);
}
```

e.g. 3 reduction – parallelize with OpenMP.

```
#define N 1000000

main()
{
    double oscillation, sum;
    int i;

    oscillation = 1.0;
    sum = 0.0
    for (i = 0; i < N; i++) {
        sum += oscillation/(2*i+1);
        oscillation   = -oscillation;
    }
    printf("%f₩n", 4.0*sum);
}
```

ex. reduction – rewrite using critical or atomic(sum.c).

```
#define N 12
int printf(char *, ...);

main()
{
    int tid, A[N], sum, i;
    omp_set_num_threads(4);

    #pragma omp parallel for
    for (i=0; i<N; i++)
        A[i] = i*2;

    sum = 0;
    #pragma omp parallel for reduction (+:sum) private(tid)
    for (i=0; i<N; i++) {
        tid = omp_get_thread_num();
        sum = sum + A[i];
        printf("%d %d %d %d₩n", tid, i, A[i], sum);
    }

    printf("%d₩n", sum);
}
```

After done, submit your program(sum.c).