1. Please implement the parallel prefix-sum algorithm according to what you learned from the class. To solve a new problem using the same idea is highly recommended.
2. A sequential program for calculating Pi is given as below. Please rewrite it into a concurrent program, and you can use either the parallel idea or the concurrent idea learned in the class. Then try to compare the efficiency between the original sequential program and your concurrent programs when the input “n” is the same. In addition to the source code, a report for the comparison is required.
3. **import** java.util.Scanner;
4. **public** **class** P9\_1 {
6. **static** **double** MontePI(**int** n) {
7. **double** PI;
8. **double** x, y;
9. **int** i, sum;
10. sum = 0;
11. **for** (i = 1; i < n; i++) {
12. x = Math.random();
13. y = Math.random();
14. **if** ((x \* x + y \* y) <= 1) {
15. sum++;
16. }
18. }
19. PI = 4.0 \* sum / n;
20. **return** PI;
22. }
24. **public** **static** **void** main(String[] args) {
25. **int** n;
26. **double** PI;
27. System.out.println("蒙特卡洛概率算法计算圆周率:");
28. Scanner input = **new** Scanner(System.in);
29. System.out.println("输入点的数量：");
30. n = input.nextInt();
31. PI = MontePI(n);
32. System.out.println("PI="+PI);
34. }
36. }

3．Please use the lock interface provided by JDK to rewrite the shared counter used in your previous homework. Then try to compare the results with the synchronized blocks.

4. Please prove that the parallelism of the merge sort algorithm is O(n/log2n)

5.Finish the exercises 11 in Section 2.10 of the text book (The art of multiprocessor programming):