Question 1

Rename.java (portion)

```
public Result rename() {

    boolean search = true;
    int down = 0, right = 0, id = -1;

    while (search) {

        Splitter splitter = null;

        try {
            splitter = m_splitters[down][right];
        } catch (Exception e) {
            return null;
        }

        if (splitter == null) {
            return null;
        }

        Direction direction = splitter.getDirection(Thread.currentThread()
                .getId());

        switch (direction) {
        case DOWN: {
            down++;
```

```
                break;
            }
            case RIGHT: {
                right++;
                break;
            }
            case STOP: {
                id = getId(down, right, m_range);
                search = false;

                break;
            }
            }
        }

        return new Result(down, right, id);
    }

    public static int getId(int down, int right, int range) {

        int y = down * (down + 1) / 2 + 1;
        int x = 0;

        // Arithmetic series
        if (right > 0) {
            int dx = down + 1;
            int ex = dx + right;
            int sx = dx * (dx + 1) / 2;
            x = ex * (ex + 1) / 2 - sx;
        }

        return x + y;
    }
```

Splitter.java (portion)

```
    public Direction getDirection(long pid) {

        m_pid.set(pid);

        if (m_stopped.get()) {
            return Direction.RIGHT;
        } else {
            m_stopped.set(true);
            if (m_pid.get() == pid) {
                return Direction.STOP;
            } else {
                return Direction.DOWN;
            }
        }
    }

    public void release() {
        m_stopped.set(false);
    }
```

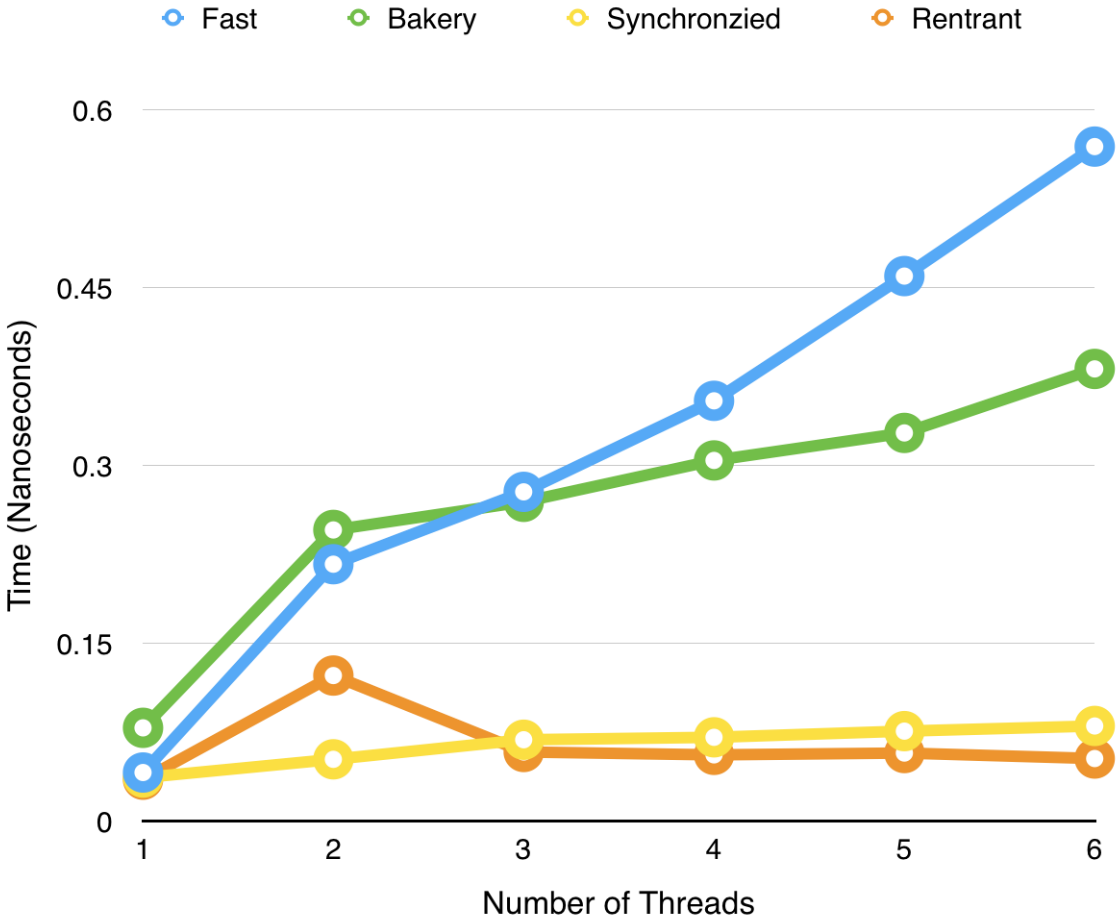
Visual

Question 2

If a read occurs concurrently with a write, the Bakery algorithm would fail. Consider the case of two threads A and B. An example of this is as follows:

Thread A	**Thread B**	
Enter CS	Waiting in while-loop	label[A] = 1; label[B] = 2;
Exits CS	OS halts	label[A] = 0;
Lock called		label[A] = 1; label[B] = 2;
Update label[A] = 3	Reads wrong label. Enter CS.	label[A] = 55; //Thread B reads Thread A wrong
Enters CS	Enters CS	label[A] = 3; label[B] = 2;

Question 3


Fast
Bakery
Synchronzied
Rentrant
0.6
0.45
0.3
0.15
0
Time (Nanoseconds)
1
2
3
4
5
6
Number of Threads