Esercizio 3

Si descriva (i) il funzionamento della seguente applicazione Java, (ii) l'output che può produrre, e (iii) se l'applicazione termina.

```
public class Prova3_20210713_A {
    static Semaphore semA = new Semaphore(2), semB = new Semaphore(0);
    static int count = 0;

    static class A extends Thread {
        public void run() {
            try {
                while(true) {
                    semA.acquire();
                    System.out.print("A");
                    semB.release();
                }
            } catch (InterruptedException e) {e.printStackTrace();}
        }
    }
    static class B extends Thread {
        public void run() {
            try {
                while(true) {
                    semB.acquire(2);
                    TimeUnit.SECONDS.sleep(3);
                    count++;
                    System.out.print("B("+count+")");
                    semA.release(2);
                }
            } catch (InterruptedException e) {e.printStackTrace();}
        }
    }
    public static void main(String[] args) {
        new A().start(); new B().start();
    }
}
```

Esercizio 4: Traccia Funivia

- Si consideri una funivia che permette di spostare i turisti da un piccolo paese fino alla cima della montagna. La funivia può essere occupata da 6 **turisti che sono arrivati lì a piedi** o da 3 **turisti in bici** (i tre posti rimanenti sono occupati dalle bici).
- La funivia è guidata da un **pilota** che continuamente sale e scende dalla montagna. Il pilota una volta arrivato a valle fa entrare nella funivia un gruppo di turisti a piedi oppure un gruppo di turisti in bici. Non potranno mai salire sia turisti a piedi sia turisti in bici. Il pilota usa una politica *round-robin*: fa salire un gruppo di 6 turisti a piedi, poi un gruppo di 3 turisti in bici e così via.
- La funivia parte solo dopo aver raggiunto il pieno carico (o 6 turisti a piedi o 3 turisti in bici) impiegando 5 minuti per giungere in cima. Il pilota, una volta arrivato in cima, lascia i turisti e ritorna con la funivia vuota a valle impiegando 2 minuti (i turisti scenderanno a piedi o in bici dalla montagna a valle).

Esercizio 4: Traccia Funivia

- Si modelli il sistema descritto in Java, dove il *pilota* e i *turisti* sono dei thread che interagiscono tramite un oggetto *Funivia* che espone (almeno) i seguenti metodi:
  - **void pilotaStart:** il pilota ha portato la funivia a valle e attende che salgano i turisti secondo l'ordine specificato sopra. Quando la funivia è piena, il pilota blocca gli accessi e inizia il viaggio.
  - **void pilotaEnd**: Il pilota è arrivato in cima, stampa l'ID dei turisti presenti nella funivia e il loro tipo e dopo permette ai turisti di scendere dalla funivia. Subito dopo inizia il ritorno a valle.
  - **void turistaSali (int t):** il turista di tipo **t** (***0*** turista a piedi, ***1*** turista in bici) è pronto per salire in montagna. Il turista viene sospeso fin quando non occupa un posto all'interno della funivia. I turisti in fila vengono risvegliati secondo un ordine causale.
  - **void turistaScendi (int t)**: permette al turista di tipo **t** di scendere dalla funivia.
- Si implementi la classe Funivia (astratta), Turista e Pilota, e una soluzione che riproduca il funzionamento del problema sopra descritto utilizzando la classe **Semaphore** (usare solo i metodi *acquire* e *release*) del package **java.util.concurrent**. Si scriva anche un ***main*** d'esempio che faccia uso di questa soluzione. Il ***main***, dopo aver definito la Funivia e avviato il pilota, esegue 18 turisti a piedi e 9 turisti in bici.