Task 1

An adventure park has several tracks. The following program simulates passing the tracks.

```java
public class Track {
    private char trackLabel;
    private int timeToPass;
    public Track(char trackLabel, int timeToPass) {
        this.timeToPass = timeToPass;
        this.trackLabel = trackLabel;
    }

    public void passTrack(Visitor visitor) throws InterruptedException {
        System.out.println(visitor.getNimi() + " started the track " + trackLabel + " " + ((System.currentTimeMillis() - visitor.getStartTime()) / 1000));
        Thread.sleep(1000 * timeToPass);
        System.out.println(visitor.getNimi() + " completed the track " + trackLabel + " " + (System.currentTimeMillis() - visitor.getStartTime()) / 1000);
    }
}

public class Visitor implements Runnable {
    private String visitorName;
    private boolean isFromBeginning;
    private Track[] tracks;
    private long startTime;
    public Visitor(String visitorName, Track[] tracks, boolean isFromBeginning) {
        this.visitorName = visitorName;
        this.tracks = tracks;
        this.isFromBeginning = isFromBeginning;
    }
    public String getNimi() {
        return visitorName;
    }
    public long getStartTime() {
        return startTime;
    }
    public void run() {
        this.startTime = System.currentTimeMillis();
        try {
            if (isFromBeginning) {
                for (int i = 0; i < tracks.length; i++)
                    tracks[i].passTrack(this);
            } else {
                for (int i = tracks.length - 1; i >= 0; i--)
                    tracks[i].passTrack(this);
            }
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}

public class AdventurePark {
    public static void main(String[] args) {
        Track[] tracks = {new Track('A', 10), new Track('B', 20)};
        Visitor visitor1 = new Visitor("Rasmus", tracks, true);
        Visitor visitor2 = new Visitor("Grete", tracks, false);
        Thread t1 = new Thread(visitor1);
        Thread t2 = new Thread(visitor2);
        t1.start();
        t2.start();
    }
}
```
Task 1.1
What is the output of the program? Why?

Task 1.2
What is the output of the program if the *passTrack* method is synchronized? Why?

Info about the **Thread class**

```java
static void sleep(long millis)
    Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds, subject to the precision and accuracy of system timers and schedulers.
```

Info about the **System class**

```java
static long currentTimeMillis()
    Returns the current time in milliseconds.
```
Task 2

The aim of the program is to simulate children playing with toys in the kids’ room. The program has to meet the following requirements:

- Several children can play in the kid’s room at the same time.
- A child can take only one toy at a time from the toy’s shelf.
- One toy can be in one kid’s hands at a time.
- A child prefers a toy whose name is the longest.
- When a child gets bored playing with the toy, the child puts the toy back to the shelf.

Unfortunately, the following program has not meet all the requirements. The following statements help find mistakes.

1. A child cannot play in the kids’ room because his thread can be started before a kids’ room is created.

2. The run method of the Child class has not been invoked.

3. The pickToy method of the KidsRoom must be synchronized.

4. The returnToy method of the KidsRoom class must be synchronized.

5. The run method of the Child class must be synchronized.

6. The instances of the Child class cannot have parameters because the Child class implements the Runnable interface.

7. The second child will never start playing because the first kid’s activity is in the infinite loop.

8. The child1 variable is created incorrectly – its datatype is Runnable, which is invalid.

9. The child2 variable is incorrect – its datatype is Child, instead of Runnable.
public class Child implements Runnable {
    private KidsRoom kidsRoom;
    private String childName;

    public Child(KidsRoom kidsRoom, String childName) {
        this.kidsRoom = kidsRoom;
        this.childName = childName;
    }

    public void run() {
        while (true) {
            String toy = kidsRoom.pickToy();
            System.out.println(childName + " plays with " + toy);
            kidsRoom.returnToy(toy);
        }
    }
}

public class KidsRoom {
    private String[] shelf;

    public KidsRoom(String[] toys) {
        this.shelf = toys;
    }

    public String pickToy() {
        int best = -1;
        for (int i = 0; i < shelf.length; i++) {
            if (shelf[i] != null && (best == -1) ||
                shelf[i].length() > shelf[best].length()) {
                best = i;
            }
        }
        String choice = shelf[best];
        shelf[best] = null;
        return choice;
    }

    public void returnToy(String toy) {
        for (int i = 0; i < shelf.length; i++) {
            if (shelf[i] == null) {
                shelf[i] = toy;
                break;
            }
        }
    }
}

public static void main(String[] args) {
    KidsRoom kidsRoom = new KidsRoom(new String[]{"train", "puzzle", "doll"});
    Runnable child1 = new Child(kidsRoom, "Volli");
    Child child2 = new Child(kidsRoom, "Martin");
    Thread t1 = new Thread(child1);
    Thread t2 = new Thread(child2);
    t1.start();
    t2.start();
}