1. A program is to be written to simulate a ski hill. The class `Skier` represents a single skier in this simulation. A `Skier` has a method `time`() that returns its current time (the time that it reaches the trail, later the time that it will reach the bottom of the trail) and another method, `setTime` that changes that time. A third `Skier` method `skillLevel`() returns the integer skill level of the skier. `Skier` implements the `Comparable` interface so that skiers are ordered by their current time. The class `Skier` is shown below.

```
public class Skier implements Comparable
{
   public double time()
   { /* implementation not shown */ }

   public void setTime(double time)
   { /* implementation not shown */ }

   public int skillLevel()
   { /* implementation not shown */ }

   public int compareTo(Object other)
   {
      Skier otherSkier = (Skier)other;
      return (int)(time() - otherSkier.time());
   }

   // constructors and other methods not shown
}
```

The class `SkiTrail` represents a single ski trail that a skier can take. `SkiTrail` has the responsibility of adding a skier to the trail, using its method `addSkier`. When a skier is added, the `SkiTrail` determines how long it will take the skier to reach the bottom of the trail and sets the skier's current time to be its old current time (when it arrived at the trail) plus the time it takes to go down the trail. The `SkiTrail` has a priority queue that it uses to store the skiers on the trail (so that they will be removed from the priority queue in the order of their current times – the times they arrive at the bottom of the trail). The class `SkiTrail` also has a method `skiersDownAtTime` that takes a double, `clockTime`, as parameter and returns a `TreeSet` of all those skiers that arrive at the bottom of the trail before `clockTime`. A partial declaration of `SkiTrail` is shown below.

```
public class SkiTrail
{
  private PriorityQueue<Skier> skierPQ;  // initialized as empty
                                         //   in constructor
  private double length;                 // initialized in constructor

  // constructor not shown

  public void addSkier(Skier sk)
  { /* to be completed as part (a) */ }

  public TreeSet<Skier> skiersDownAtTime(double clockTime)
  { /* to be completed as part (b) */ }
}
```

(a) Complete the `SkiTrail` method `addSkier`. This method should calculate the time it takes for the skier to go down the trail by dividing the length of the trail by the product of the skier skill level and a random `double` between 1.0 and 2.0. The method should then set the skiers current time to be its previous current time plus the time to get down the hill. Finally the method `addSkier` should put the skier into the priority queue `skierPQ`.

Complete method `addSkier` below.

```
public void addSkier(Skier sk)
```

(b) Complete the `SkiTrail` method `skiersDownAtTime` below. This method should remove all the skiers in the priority queue `skierPQ` whose current times are smaller than the parameter `clockTime` and return them in a `TreeSet`.

Complete method `skiersDownAtTime` below.

```
public TreeSet skiersDownAtTime(double clockTime)
```