2011 Paper 3 Question 7

Further Java

In this question you will need to fill in missing parts of a Java program. You may ignore any exception handling and will not be penalised for minor syntactic errors.

You are provided with a class `Eval`:

```
public class Eval {
  public static int f(Record r) { ... }
}
```

(*a*) Add another method `Integer maxf(Iterator<Record> it)` to the class `Eval`. Your method should return the maximum value computed by `f` for every `Record` returned by the iterator or `null` if there are no records available. The relevant portion of the `Iterator` interface is as follows:

```
interface Iterator<T> {
  // return true if there are more values available
  public boolean hasNext();

  // return the available value and advance to the next one
  public T next();
}
```

[5 marks]

(*b*) Complete the methods `run()` and `join()` in the following abstract class. You may add additional fields or methods if you wish.

```
abstract class Joinable implements Runnable {
  abstract void exec();
  final public void run() {
    // ... call the exec() method ...
  }
  void join() throws InterruptedException {
    // block the calling thread until exec() completes in run()
  }
}
```

[7 marks]

(*c*) Provide a method `Integer parmaxf(Iterator<Record> it, int n)` which is functionally equivalent to `Eval.maxf`, except that it should create `n` parallel threads of execution to speed up the calculation of the result. You may assume that `Iterator<Record>` is thread-safe. You may find it helpful to subclass the `Joinable` class. [8 marks]