Running Trails Write-up
Daniel Kopeloff

A) Describe the optimal substructure/recurrence that would lead to a recursive solution

We could take the midpoint of the possible speeds and then test the athlete. We would perform this type of average until we get only two speeds left. Once we achieved the to two possible speeds then we know that we need to only conduct one more test since it is either the lower of the two speeds or the higher one.

B) Code your recursive solution under runTrialsRecur(int possibleSpeeds, int days). If your recursive function runs forever, in order for grading to happen quickly please comment out the code progress you made instead.

```
// Do not change the parameters!
public int runTrialsRecur(int possibleSpeeds, int days) {
  int minTests = 0;

  // Your code here
//    System.out.println("Speeds " + possibleSpeeds);
//    System.out.println("Days Remaining " + days);

 if((days == 1 || days == 2) || possibleSpeeds <= 2 ) {
      minTests++ ;
      return minTests;
  }
  else {
  minTests++;
  minTests = minTests + runTrialsRecur(possibleSpeeds / 2 , days - 2);
  //System.out.println(minTests);
  return minTests;
  }


}
```

C) Draw recurrence tree for given (# speeds = 6, # days = 2)

n = 6 **m = 2**

T(1) = 1 T(n,m) = T(n/2,m-2) + C(n,m)

Tree

T(n,m) C(n,m)

T(n/2,m-2) n/2,m-2

D) **How many distinct subproblems do you end up with given 6 speeds and 2 days?**

1

E) **How many distinct subproblems for N speeds and M days?**

Take the celling of M/2

F) **Describe how you would memoize runTrialsRecur.**

I would create a tree that every level signifies a day then add and entry on that level for that N possible speed.

G)