*** OUTPUT **********

Note: Combination.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Welcome to Online IDE!! Happy Coding :)
COMBINATIONS (WITH REPLACEMENT)
C^R(4,4) = (n+r-1)! / r!(n-1)!
C^R(4,4) = 7! / 4!(3)!
35
Combinations: 35 (This will be maximum limit for set size)

Size of list: 4
random number: 2
This is number in list: 76
Size of list: 4
random number: 1
This is number in list: 7
Size of list: 4
random number: 3
This is number in list: 415
Size of list: 4
random number: 0
This is number in list: 10
This will be stored in set: 76741510
set size: 35

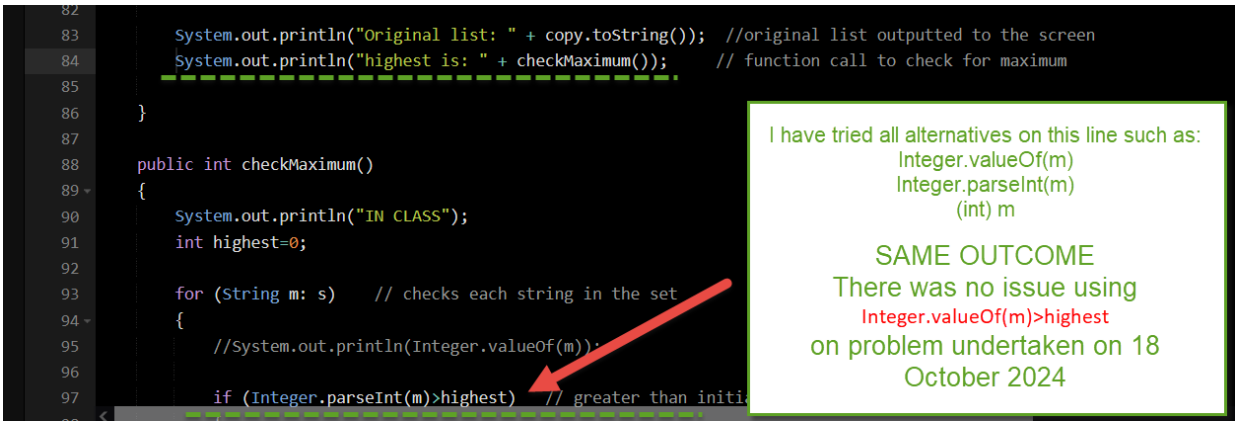
Number cycles: 38 // This has finished executed similar cycle to 35
Original list: [10, 7, 76, 415]

RUNTIME ERROR AS BELOW:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "415415415415"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
    at java.base/java.lang.Integer.parseInt(Integer.java:652)
    at java.base/java.lang.Integer.parseInt(Integer.java:770)
    at largestNumber.checkMaximum(Combination.java:97)
    at largestNumber.<init>(Combination.java:84)
    at Combination.main(Combination.java:124)


** Process exited - Return Code: 1 **
```

This points to following code:



I have also examined the first number in the set that it processes and there are no characters that would fail to parse from String to Int. The output below also does not seem to be the first entry!

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "415415415415"
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "41576415415"
```

/
*** CODE **
//FAILING!!! - COMBINATION WITHOUT REPLACEMENT

/*
Online Java - IDE, Code Editor, Compiler

Online Java is a quick and easy tool that helps you to build, compile, test your programs online.
*/

// This has been created to ensure I can utilize any random functions more efficiently.
// It is a creation of the combinations with replacement calculator.
// It has used techniques I learnt including recursion and also memoization to speed up

```
execution.
// I will incorporate this into Java applications I created previously..

import java.math.*;
import java.util.*;


class largestNumber
{
   long combinations;
   int count;
   String temp;

   Set <String> s = new HashSet <>();   // this will store the  combinations

   List <Integer> lst = new ArrayList<>(Arrays.asList(10,7,76,415));  // this is list of numbers. In
future, it will be chosen to use other data

   List <Integer> copy = new ArrayList<>(lst);  //keeps a copy    // this list keeps a copy since
during program execution the top list is modified


   public largestNumber(long combinations)
   {

      this.combinations=combinations;
      int randomNumber;    // random  number generated
      int numberArray;     // value at index of randomNumber in the set
      int counter=0;

      Random rand = new Random();  // generates random number

      System.out.println("Combinations: " + combinations +"\n");   //number combinations
without replacement


      do
      {

         temp="";   // this is used concatenation of value before it is stored in the set....
         counter=0;

         do
         {
            System.out.println("Size of list: " + lst.size());        //size list

            randomNumber = rand.nextInt(lst.size());               //random number between  0 -
(size list-1)
            System.out.println("random number: " + (randomNumber));

            numberArray = lst.get(randomNumber);   //gets number from the list
            System.out.println("This is number in list: " + numberArray);    //corresponding value
in list
```

```
            temp =  temp + Integer.toString(numberArray);    // concatenating the values

            counter++;

            //lst.remove(randomNumber);     // this is important step.. it removes value at this
index from list... enforce combination without replacment

            //System.out.println("value of i: " + i + "\n");

        } while (counter<lst.size());          // it will keep processing while this is true....   while
list is not empty

        s.add(temp);    // adding the value to the set.
        System.out.println("This will be stored in set: " + temp);
        System.out.println("set size: " + s.size() +"\n");   //once this has incremented, the set
has grown...


     count++;   // this is indication of a cycle.. It is aiming to be close to combinations...


     } while (s.size()<combinations);    // this is 1 less due to 0 indexing of the set.


     System.out.println("Number cycles: " + count);

     System.out.println("Original list: " + copy.toString());  //original list outputted to the
screen
     System.out.println("highest is: " + checkMaximum());     // function call to check for
maximum

  }

  public int checkMaximum()
  {
     System.out.println("IN CLASS");
     int highest=0;

     for (String m: s)    // checks each string in the set
     {
        //System.out.println(Integer.valueOf(m));

        // StringBuilder sb = new StringBuilder(m);   // also tried StringBuilder but this also
failed....


        if (Integer.parseInt(m)>highest)   // greater than initial 0.. Strings are converted back
to //integer
        {
          highest=Integer.valueOf(m);      // it will store value
        }
```

```
      }

      return highest;
   }
}


public class Combination
{
   public static void main(String[] args) {
      System.out.println("Welcome to Online IDE!! Happy Coding :)");

      int originalNumber=5;
      int n=originalNumber;
      int r =6;
      Map <Integer, Long> m = new HashMap<>();
      System.out.println("***COMBINATIONS*** (WITH REPLACEMENT)");

      System.out.println("C^R(" + n+","+r+") = " + "(n+r-1)! / r!(n-1)!");
      System.out.println("C^R(" + n+","+r+") = " + (n+r-1)+ "!" + " / " + r+"!"+"("+(n-1)+")!");
      System.out.println(Combinations (n,r,originalNumber, m));

      largestNumber ln = new largestNumber(Combinations (n,r,originalNumber, m));

   }

   public static long Combinations (int n,  int r, int originalNumber, Map factorialResults)
   {

      // n are objects
      // r is sample

     /*
     ***CALCULATION***
      (n+r-1)! / r!(n-1)!

      */

      long result=0;
      int denominator1;   //denominator split two parts since there are two factorial
calculations
      int denominator2;   //denominator split two parts since there are two factorial
calculations
      int Numerator=n+r-1;  // Numerator

      if (Numerator>=1)  // this will ensure that all factorials as low as 1! are processed
      {
         //System.out.println("value of n: " + Numerator);
         // EXAMPLE

         // C^R (5,6)   =   (5+6-1)!   /   6! (5-1)!     =    3628800 / (6! * 4!) =  3628800 / 720 * 24 =
210
```

```
    result = ((n+r-1)* (Combinations (n-1, r,originalNumber, factorialResults)));   // this
completes factorial for numerator

    factorialResults.put(Numerator,result);   //result stored in the Map
    //factorialResults.put(n-1,result);   //result stored in the Map
    //System.out.println("getting result back out: " + (Numerator) + "   " +
factorialResults.get(n+r-1));


    if (n==originalNumber)   // this will occur once
    {

      denominator1 = r;       // r sample size has not changed

      denominator2 = originalNumber-1;    // originalNumber required since n has reduced
as part of the recursive calls


    // this is using the Java Memoization technique to ensure the factorial outcome is not
calculated again, to save program cycles.
    // since the returns are done in reverse order.... n = 1 is processed first and n=6 last...
Hence in practice
    // there will be entry in Map for all factorials, ready for the denominator..

    //n+r-1        r=6     n = 3 or 2 or 1    so recursive values going in set are    8! ,  7!,  6!
factorial
    // but the put method for the set would recursively call and populate others up to 1!

      if (factorialResults.containsKey(denominator1) &&
factorialResults.containsKey(denominator2))

      {
        //System.out.println("here");
        //System.out.println("This is exact value of factorial  " + (denominator1) + " : " +
factorialResults.get(denominator1));
        //System.out.println("This is exact value of factorial  " + (denominator2) + " : " +
factorialResults.get(denominator2));

        return result / ((long) factorialResults.get(denominator1) *
(long)factorialResults.get(denominator2));
      }

    }

      return result;

    }

      return 1;    // it will reach here only when condition not met (Numerator>=1)
    }

  }
```