ATTEMPT 1

```
/*
Online Java - IDE, Code Editor, Compiler

Online Java is a quick and easy tool that helps you to build, compile, test your programs online.
*/
import java.util.*;
public class Main
{
   static String sequence;
   static int n=1;
   //final static String str = "123124125";
   //final static String str = "101112131415";
   //final static String str = "32332432536";
   //final static String str = "326325324323";
   //final static String str = "666667";

   final static String strBackup=str;
   static StringJoiner sj = new StringJoiner(",");
   static StringBuilder ascNums = new StringBuilder("");
   static Boolean alternatingBlocks = false;


   public static void main(String[] args)
   {
      System.out.println("Welcome to Online IDE!! Happy Coding :)");
      System.out.println("Number to be examined: " + str);
      ascending(str);
   }

   public static boolean ascending(String str)
   {
      String blockBefore="";
      System.out.println("This is the block size: " + n);
```

```
if((str.length()%n==0))
{
  if (!(str.length()<(2*n)))
  {
    System.out.println("There are minimum of two blocks of size: " + n);
    String block = str.substring(str.length()-n, str.length());

    try
    {
      blockBefore = str.substring((str.length() - (2*n)),(str.length()-n));

      System.out.println("This is block: " + block);
      System.out.println("This is block before: " + blockBefore);

     //This appears to be first line where the logic should be modified
    // But we know immediately it is not case of introducing  || since the blocks are either ascending
      or the blocks are either descending
     //Also need to refrain from copying the entire block of code again
     //It points towards putting a boolean in the associated else statement
      //trying something as similar will put a lock on the loop until it has performed the first execution
     //so I need to evaluate the if statement before the entrant point also.

     if (state==”ascending” && executeOnce)
     {
      if ((Long.valueOf(blockBefore))<Long.valueOf(block))

      {
       state=”ascending”;
        executeOnce=true;
        String blockBeforeBlock = str.substring((str.length() - (3*n)),(str.length()-(2*n)));
        String currentBlock = str.substring(str.length()-n,str.length());

        System.out.println("Performing recursion, examining block " + "("+blockBefore+")" +" with
block before it" + "("+blockBeforeBlock+")");
        sj=new StringJoiner(",");

        if (ascNums.length()==0)
```

```
            {
              sj.add(blockBeforeBlock);
              sj.add(blockBefore);
              sj.add(block);
            }
            else
            {
              sj.add(blockBeforeBlock);
            }

            sequence = sj.toString()+",";

            ascNums.insert(0,sequence);

            System.out.println("*****THIS IS THE SEQUENCE: " + ascNums);

            ascending(str.substring(0,(str.length()-n)));
          }

        }

        else
          {
            n=n+1;
            System.out.println("Performing recursion, restoring original String");
            sj=new StringJoiner(",");
            ascNums=new StringBuilder("");
            ascending(strBackup);
          }
        }
        catch (StringIndexOutOfBoundsException | ArrayIndexOutOfBoundsException e)
        {
          System.out.println(blockBefore + " are digit(s) of first block n"+"("+n+")" + " of " + strBackup);
          System.out.println(strBackup + " consists of ascending numbers of group size" +"("+n+"): " +
ascNums);
```

```
            System.exit(0);
            return true;
        }
    }
}

else
{
    n++;
    System.out.println("Value of block size has been increased to: " + n);
    sequence="";
    ascNums=new StringBuilder("");
    sj=new StringJoiner(",");

    str=strBackup;

    System.out.println("String length: " + str.length());
    System.out.println("Block size: " + n);

    if (str.length()%2==0)
    {
        if (n<(str.length()/2))
        {
            ascending(str);
        }
    }
    else
    {
        if (n<((str.length()+1)/2))
        {
            ascending(str);
        }
    }
}
```

```
      System.out.println(strBackup + " does not consist of ascending number in any block size " + n + "(n): "
+ ascNums);

      System.exit(0);

      return false;

   }

}
```

ATTEMPT 2

```
/*

Online Java - IDE, Code Editor, Compiler


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

*/

import java.util.*;

public class Main

{

   static String sequence;

   static int n=1;

   //final static String str = "123124125";

   //final static String str = "101112131415";

   //final static String str = "32332432536";

   //final static String str = "326325324323";

   //final static String str = "666667";


   final static String strBackup=str;

   static StringJoiner sj = new StringJoiner(",");

   static StringBuilder ascNums = new StringBuilder("");

   static boolean executeOnce = false;

   static String state = “”;

   static String initialState=””;


   public static void main(String[] args)

   {
```

```
    System.out.println("Welcome to Online IDE!! Happy Coding :)");
    System.out.println("Number to be examined: " + str);
    ascending(str);
  }

  public static boolean ascending(String str)
  {
    String blockBefore="";
    System.out.println("This is the block size: " + n);

    if((str.length()%n==0))
    {
      if (!(str.length()<(2*n)))
      {
        System.out.println("There are minimum of two blocks of size: " + n);
        String block = str.substring(str.length()-n, str.length());
        try
        {
          blockBefore = str.substring((str.length() - (2*n)),(str.length()-n));

          System.out.println("This is block: " + block);
          System.out.println("This is block before: " + blockBefore);

        //This appears to be first line where the logic should be modified

        if ((Long.valueOf(blockBefore))<Long.valueOf(block) /*&&
executeOnce*/)
        {
          state = “ascending”;
          if (state!=initialState)
          {
            //We know blocks have changed direction or the block values are
            //identical
           //We need to invalidate the variables so that it does not enter the
           if statement.  Or set boolean logic as below
           alternatingBlocks = true;
          }
```

```
}
else
{
 state=”descending”;
 if (state==initialState)
 {
  blockBefore = block;
  block=blockBefore;
 }
 else
 {
  alternatingBlocks=true;
 }
}
```

//There is a problem here, although logic is intertwined… There is nothing telling the code below
//to enter from the perspective of descending… It has ONLY switched the variables about if it has
//performed executeOnce
//BUT if we remove the executeOnce boolean from the above (see commented red section few lines above), it will then attempt to perform this line (if (state!=initialState)) before it even reaches the code below. It will set state=”ascending” alternatingBlocks=true;
So now, I would need to comment the altenatingBlocks in the loop below…

So once again, the logic has failed….. Time to move to attempt 3!!!

//So the only way would be to make a comparison in below if it is performing a 2nd iteration for the recursive call…

```
if ((Long.valueOf(blockBefore))<Long.valueOf(block) /*&&!alternatingBlocks*/)
{
  executeOnce=true;

    //this has to capture the first state since it dictates if traversing
      ascending or descending
      initialState = state;

    String blockBeforeBlock = str.substring((str.length() - (3*n)),(str.length()-(2*n)));

    String currentBlock = str.substring(str.length()-n,str.length());

    System.out.println("Performing recursion, examining block " + "("+blockBefore+")" +" with block before it" + "("+blockBeforeBlock+")");

    sj=new StringJoiner(",");
```

```
            if (ascNums.length()==0)
            {
               sj.add(blockBeforeBlock);
               sj.add(blockBefore);
               sj.add(block);
            }
            else
            {
               sj.add(blockBeforeBlock);
            }

            sequence = sj.toString()+",";

            ascNums.insert(0,sequence);

            System.out.println("*****THIS IS THE SEQUENCE: " + ascNums);

            ascending(str.substring(0,(str.length()-n)));
          //executeOnce=false;
          }


else
        {
            n=n+1;
            System.out.println("Performing recursion, restoring original String");
            sj=new StringJoiner(",");
            ascNums=new StringBuilder("");
            ascending(strBackup);
          }
        }
        catch (StringIndexOutOfBoundsException | ArrayIndexOutOfBoundsException e)
        {
           System.out.println(blockBefore + " are digit(s) of first block n"+"("+n+")" + " of " + strBackup);
```

```
            System.out.println(strBackup + " consists of ascending numbers of group size" +"("+n+"): " +
ascNums);


            System.exit(0);
            return true;
          }
        }
      }

      else
      {
        n++;
        System.out.println("Value of block size has been increased to: " + n);
        sequence="";
        ascNums=new StringBuilder("");
        sj=new StringJoiner(",");

        str=strBackup;

        System.out.println("String length: " + str.length());
        System.out.println("Block size: " + n);

        if (str.length()%2==0)
        {
          if (n<(str.length()/2))
          {
            ascending(str);
          }
        }
        else
        {
          if (n<((str.length()+1)/2))
          {
            ascending(str);
          }
        }
```

```
      }


      System.out.println(strBackup + " does not consist of ascending number in any block size " + n + "(n): "
+ ascNums);
      System.exit(0);
      return false;
    }
}
```

I am not content at ALL. There appears to be a dependency between both scenarios
THE CODE BELOW, IF I CHOOSE TO TRY AGAIN!

ATTEMPT 3

```
/*
Online Java - IDE, Code Editor, Compiler

Online Java is a quick and easy tool that helps you to build, compile, test your programs online.
*/
import java.util.*;
public class Main
{
   static String sequence;
   static int n=1;
   //final static String str = "123124125";
   //final static String str = "101112131415";
   //final static String str = "32332432536";
   //final static String str = "326325324323";
   //final static String str = "666667";

   final static String strBackup=str;
   static StringJoiner sj = new StringJoiner(",");
   static StringBuilder ascNums = new StringBuilder("");
```

```
public static void main(String[] args)
{
   System.out.println("Welcome to Online IDE!! Happy Coding :)");
   System.out.println("Number to be examined: " + str);
   ascending(str);
}

public static boolean ascending(String str)
{
   String blockBefore="";
   System.out.println("This is the block size: " + n);

   if((str.length()%n==0))
   {
      if (!(str.length()<(2*n)))
      {
         System.out.println("There are minimum of two blocks of size: " + n);
         String block = str.substring(str.length()-n, str.length());

         try
         {
            blockBefore = str.substring((str.length() - (2*n)),(str.length()-n));

            System.out.println("This is block: " + block);
            System.out.println("This is block before: " + blockBefore);

          //This appears to be first line where the logic should be modified
         // But we know immediately it is not case of introducing   || since the blocks are either ascending
            or the blocks are either descending
         //Also need to refrain from copying the entire block of code again
         //It points towards putting a boolean
            if ((Long.valueOf(blockBefore))<Long.valueOf(block))

            {
```

```
            String blockBeforeBlock = str.substring((str.length() - (3*n)),(str.length()-(2*n)));
            String currentBlock = str.substring(str.length()-n,str.length());

            System.out.println("Performing recursion, examining block " + "("+blockBefore+")" +" with block before it" + "("+blockBeforeBlock+")");
            sj=new StringJoiner(",");

            if (ascNums.length()==0)
            {
              sj.add(blockBeforeBlock);
              sj.add(blockBefore);
              sj.add(block);
            }
            else
            {
              sj.add(blockBeforeBlock);
            }

            sequence = sj.toString()+",";

            ascNums.insert(0,sequence);

            System.out.println("*****THIS IS THE SEQUENCE: " + ascNums);

            ascending(str.substring(0,(str.length()-n)));
          }
          else
          {
            n=n+1;
            System.out.println("Performing recursion, restoring original String");
            sj=new StringJoiner(",");
            ascNums=new StringBuilder("");
            ascending(strBackup);
          }
        }
        catch (StringIndexOutOfBoundsException | ArrayIndexOutOfBoundsException e)
```

```
        {
            System.out.println(blockBefore + " are digit(s) of first block n"+"("+n+")" + " of " + strBackup);
            System.out.println(strBackup + " consists of ascending numbers of group size" +"("+n+"): " +
ascNums);

            System.exit(0);
            return true;
        }
      }
    }

    else
    {
      n++;
      System.out.println("Value of block size has been increased to: " + n);
      sequence="";
      ascNums=new StringBuilder("");
      sj=new StringJoiner(",");

      str=strBackup;

      System.out.println("String length: " + str.length());
      System.out.println("Block size: " + n);

      if (str.length()%2==0)
      {
        if (n<(str.length()/2))
        {
          ascending(str);
        }
      }
      else
      {
        if (n<((str.length()+1)/2))
        {
          ascending(str);
```

```
            }
          }
        }

        System.out.println(strBackup + " does not consist of ascending number in any block size " + n + "(n): "
+ ascNums);
        System.exit(0);
        return false;
    }
}
```