```
//Jemma Tiongson
//Comp 182/L
//Prof Wang
//Lab3 - Ch.3

class Recursion{
// 1. Read Ch. 3 and write recursive codes for 1) the factorial of n, and  2) writing a
string backward. Show how these code works with results.
// 2. Implement iterative codes for 1) the factorial of n, 2) writing a string backward.
Show how these code works with results.
// 3. Discuss and compare between efficiency of recursive codes and iterative codes for
1) the factorial of n, 2) writing a string backward. Measure execution times of 1 and 2.
```

1.

```
  //recursive factorial
  public static int factorial(int n){
    //Purpose: Program will compute the factorial of an inputted value
    //Precondition: n is a positive integer and is passed through factorial as a parameter
    //Postcondition: output will be the factorial of n
    if(n == 0){
      return 1;
    }
    else{
      return n*factorial(n-1);
    }

  }
  //recursive reversed word
  public static void reversed(String word){
    //Purpose: Program will write an inputted word backwards
    //Precondition: None
    //Postcondition: The string entered will output in reverse character order
    if(word.length()>0){
      System.out.print(word.charAt(word.length()-1));
      reversed(word.substring(0,word.length()-1));
    }

  }
  //Iterative factorial
  public static int fact(int n){
    int res = 1;
    for(int i = 1; i<=n; i++){
      res *= i;
    }
```

```
    return res;
  }
  //Iterative reversed word
  public static void reversedWord(String word){
    for(int i = word.length()-1; i >= 0; i--) {
      System.out.print(word.charAt(i));
    }
  }
  public static void main(String [] args) {
    //factorial test
    double start = System.nanoTime();
    System.out.println("Recursive Factorial: " +factorial(4));
    double end = System.nanoTime();
    double difference = (end-start)/1000000;
    System.out.print("\nTakes "+difference+" for recursive factorial to finish");

    //reversed test
    double start1 = System.nanoTime();
    System.out.println("\nRecursive reversed word: ");
    reversed("jemma");
    double end1 = System.nanoTime();
    double difference1 = (end1-start1)/1000000;
    System.out.print("\nTakes "+difference1+" for recursive reverse to finish");

2. iterative factorial
    double start2 = System.nanoTime();
    System.out.print("\nIterative factorial: "+fact(4));
    double end2 = System.nanoTime();
    double difference2 = (end2-start2)/1000000;
    System.out.print("\nTakes "+difference2+" for iterative factorial to finish");

    //iterative reversed word
    double start3 = System.nanoTime();
    System.out.println("\nIterative reversed word: ");
    reversedWord("jemma");
    double end3 = System.nanoTime();
    double difference3 = (end3-start3)/1000000;
    System.out.print("\nTakes "+difference3+" for iterative reverse to finish");
  }
}
```

Resut:
Recursive Factorial: 24
Takes 0.269581 for recursive factorial to finish

Recursive reversed word:
ammej
Takes 0.192295 for recursive reverse to finish

Iterative factorial: 24
Takes 0.037903 for iterative factorial to finish

Iterative reversed word:
ammej
Takes 0.153751 for iterative reverse to finish

3. Conclusion:
Recursive solutions typically take longer than iterative solutions. For this specific example, recursive factorial and iterative factorial have a drastic difference in time. Recursive factorial takes 0.269.. nanoseconds to finish. And iterative factorial takes 0.0379 nanoseconds to finish. Making the iterative factorial much more efficient than the recursive. For recursive and the iterative reverse word method, recursive reversed word takes 0.1922.. and the iterative reversed word takes 0.1537… The difference between the two are hardly a big change. But iterative is still a bit faster than recursive. In conclusion iterative methods are more efficient than recursive.