1]Convert from hexadecimal to decimal

```
import java.util.Scanner;
public class htd{
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("Enter a hexadecimal number: ");
       String hex = sc.nextLine();
       int decimal = Integer.parseInt(hex, 16);
       System.out.println("The decimal equivalent is: " + decimal);
   }
}
```

Op:

```
Enter a hexadecimal number: 15
The decimal is: 21
```

2]Write a program that takes as input a string and removes adjacent spaces, leaving at most
one space in-a-row.

```
import java.util.Scanner;
public class RS{
   public static void main(String[] args) {
       System.out.print("Enter: ");
       Scanner sc = new Scanner(System.in);
       String input = sc.nextLine();
       String result = input.replaceAll("\\s+", "");
       System.out.println("String without spaces: " + result);
   }
}
```

Op:

```
Enter: my name tarun
String without spaces: mynametarun
```

3]Given a string, create a new string with all the consecutive duplicates removed. For example, ABBCCCCCBBAB becomes ABCBAB

```
import java.util.Scanner;
public class RD{
   public static void main(String[] args) {
       System.out.print("Enter a string: ");
       Scanner sc = new Scanner(System.in);
       String input = sc.nextLine();
       StringBuilder result = new StringBuilder();
       char prev = '\0';
       for (char c : input.toCharArray()) {
           if (c != prev) {
               result.append(c);
               prev = c;
           }
       }
       System.out.println("new string :" + result);
   }
}
```

```
Op:
Enter a string: ABBCCCCCBBAB
new string: ABCBAB
```

5. Write a function that takes as input a string and returns true if the string is a palindrome,and false otherwise. A palindrome is a string that reads the same forwards or backwards.

```
import java.util.Scanner;
public class palindrome{
   public static void main(String[] args) {
       System.out.print("Enter a string: ");
       Scanner sc = new Scanner(System.in);
       String input = sc.nextLine();
       String reversed = new StringBuilder(input).reverse().toString();
       System.out.println("palindrome?" + input.equals(reversed));
   }
}
```

```
Op:
Enter a string: malayalam
palindrome?true
```

6. Write a function that takes as input a string and returns true if the string is a Watson-Crick complemented palindrome, and false otherwise. A Watson-Crick complemented palindrome is a DNA string that is equal to the complement (A-T, C-G) of its reverse

```
import java.util.Scanner;

public class WC{
   public static void main(String[] args) {
       System.out.print("Enter DNA: ");
       Scanner sc = new Scanner(System.in);
       String input = sc.nextLine();
       System.out.println("palindrome? " + wcp(input));
   }

   public static boolean wcp(String s) {
       int n = s.length();
       for (int i = 0; i < n; i++) {
           char start = s.charAt(i);
           char end = s.charAt(n - 1 - i);

           if ((start == 'A' && end != 'T') ||
               (start == 'T' && end != 'A') ||
               (start == 'C' && end != 'G') ||
               (start == 'G' && end != 'C')) {
               return false;
           }
       }
       return true;
   }
}
```

```
Op:
Enter DNA: atgc
palindrome? true
```

7. Write a function that takes as input a DNA string of A, C, G, and T characters and returns
the string in reverse order with all of characters replaced by their complements. For example, if the input is ACGGAT, then return ATCCGT.

```
import java.util.Scanner;
public class DNAreverse {
   public static void main(String[] args) {
       System.out.print("Enter DNA: ");
       Scanner sc = new Scanner(System.in);
       String input = sc.nextLine();
       System.out.println("new dna:" + reverseC(input));
   }
```

```
    public static String reverseC(String s) {
        StringBuilder result = new StringBuilder();
        for (char c : s.toCharArray()) {
            switch (c) {
                case 'A' -> result.append('T');
                case 'T' -> result.append('A');
                case 'C' -> result.append('G');
                case 'G' -> result.append('C');
            }
        }
        return result.reverse().toString();
    }
}
```

```
Op:
Enter DNA: ATCGTA
new dna:TACGAT

Enter DNA: atcgta
new dna:
```

9. Write a data type TreeString.java that represents an immutable string using a binary tree.

It should support concatenation in constant time, and printing out the string in time proportional to the number of characters.

```
public class TS{
    public final String val;
    public TS(String val) {
        this.val = val;
    }
    public TS(TS left, TS right) {
        this.val = left.toString() + " " + right.toString();
    }
    @Override
    public String toString() {
        return val;
    }
    public static void main(String[] args) {
        TS s1 = new TS("heeeeeeeello");
        TS s2 = new TS("jii");
        TS s3 = new TS(s1, s2);
        System.out.println("Concatenation: " + s3);
    }
}
```

```
Op:
Concatenation: heeeeeeeello jii
```

10. In DNA sequence analysis, a complemented palindrome is a string equal to its reverse
complement. Adenine (A) and Thymine (T) are complements, as are Cytosine (C) and
Guanine (G). For example, ACGGT is a complement palindrome. Such sequences act as
transcription-binding sites and are associated with gene amplification and genetic
instability. Given a text input of N characters, find the longest complemented palindrome
that is a substring of the text. For example, if the text is GACACGGTTTTA then the
longest complemented palindrome is ACGGT. Hint: consider each letter as the center of a
possible palindrome of odd length, then consider each pair of letters as the center of a
possible palindrome of even length.

```
public class Longest{
    public static void main(String[] args) {
        String input = "GACACGTTTTTT";
        System.out.println(longest(input));
    }

    public static String longest(String text) {
        int maxLen = 0, start = 0;
        for (int i = 0; i < text.length(); i++) {
            for (int j = i; j < text.length(); j++) {
                String sub = text.substring(i, j + 1);
                if (WCP(sub) && sub.length() > maxLen) {
                    maxLen = sub.length();
                    start = i;
                }
            }
        }
        return text.substring(start, start + maxLen);
    }

    public static boolean WCP(String s) {
        StringBuilder complement = new StringBuilder();
        for (char c : s.toCharArray()) {
            switch (c) {
                case 'A' -> complement.append('T');
                case 'T' -> complement.append('A');
                case 'C' -> complement.append('G');
                case 'G' -> complement.append('C');
            }
        }
        return s.equals(complement.reverse().toString());
    }
}
```

Op:

```
ACGT
```

11. Program to demonstrate Package thought in the class.

```
public class Package{
    public static void main(String[] args) {
        System.out.println("This is package ig...");
    }
}
```

Op:

```
This is package ig…
```

4]

```
public class mystery {
    public static void main(String[] args) {
        System.out.println("The mystery output is: " + mystery(5));
    }
    public static String mystery(int N) {
        String s = "";
        while (N > 0) {
            if (N % 2 == 1) s = s + "x";
            else s = s + s;
            N = N / 2;
        }
        return s;
    }
}
```

```
Op:
The mystery output is: xxx
```

8]

```
public class MysteryStrings {
   public static void main(String[] args) {
       System.out.println("Mystery output: " + mystery("abcd", "efgh"));
   }
   public static String mystery(String s, String t) {
       int N = s.length();
       if (N <= 1) return s + t;
       String a = mystery(s.substring(0, N / 2), t.substring(0, N / 2));
       String b = mystery(s.substring(N / 2), t.substring(N / 2));
       return a + b;
   }
}
```

```
Op:
Mystery output: aebfcgdh
```