

Code : 1

```
package Topic_09_RecursionWithArrayList;
```

```
import java.util.ArrayList;
```

```
import java.util.Iterator;
```

```
import java.util.Scanner;
```

```
public class A_GetSubSequence {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String str = s.nextLine();
        ArrayList<String> res = gss(str);
        System.out.println(res);
    }

    public static ArrayList<String> gss(String str) {
        if (str.length() == 0) {
            ArrayList<String> res = new ArrayList<String>();
            res.add("");
            return res;
        }
        String charAtZero = String.valueOf(str.charAt(0));
        String ros = str.substring(1);
        ArrayList<String> result = gss(ros);
        ArrayList<String> newResult = new ArrayList<String>();
        for (int i = 0; i < result.size(); i++) {
            newResult.add(result.get(i));
        }
        for (int i = 0; i < result.size(); i++) {
            newResult.add(charAtZero + result.get(i));
        }
        return newResult;
    }
}
/*
Sample Input
abc
Sample Output
[, c, b, bc, a, ac, ab, abc]
*/
```

Code : 2

```
package Topic_09_RecursionWithArrayList;
```

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
```

```
public class B_GetKpc {
    //      0   1   2   3   4   5   6   7   8   9
    static String[] arr = { ".", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tu", "vw", "yz" };

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String str = s.nextLine();
        ArrayList<String> res = getKpc(str);
        System.out.println(res);
    }

    public static ArrayList<String> getKpc(String str) {
        if (str.length() == 0) {
            ArrayList<String> res = new ArrayList<>();
            res.add("");
            return res;
        }

        String charAtZero = String.valueOf(str.charAt(0));
        String ros = str.substring(1);

        ArrayList<String> result = getKpc(ros);
        ArrayList<String> newResult = new ArrayList<>();

        String getKeyPad = arr[Integer.valueOf(charAtZero)];
        for (int i = 0; i < getKeyPad.length(); i++) {
            for (String s : result) {
                newResult.add(String.valueOf(getKeyPad.charAt(i) + s));
            }
        }
        return newResult;
    }
}

/*

Sample Input
78
Sample Output
[tv, tw, tx, uv, uw, ux]

*/
```

Code : 3

```
package Topic_09_RecursionWithArrayList;
```

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
```

```
public class C_GetStairPaths {
    public static void main(String[] args) throws Exception {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        System.out.println(getStairPaths(n));
    }

    public static ArrayList<String> getStairPaths(int n) {
        if (n <= 0) {
            ArrayList<String> res = new ArrayList<String>();
            if (n == 0) {
                res.add("");
            }
            return res;
        }
        ArrayList<String> pathOne = getStairPaths(n - 1);
        ArrayList<String> pathTwo = getStairPaths(n - 2);
        ArrayList<String> pathThree = getStairPaths(n - 3);
        ArrayList<String> oneRes = new ArrayList<String>();
        ArrayList<String> twoRes = new ArrayList<String>();
        ArrayList<String> threeRes = new ArrayList<String>();

        ArrayList<String> paths = new ArrayList<String>();

        for (String s : pathOne) {
            paths.add(1 + s);
        }
        for (String s : pathTwo) {
            paths.add(2 + s);
        }
        for (String s : pathThree) {
            paths.add(3 + s);
        }
        return paths;
    }
}
/*
```

Sample Output for n=3

[111, 12, 21, 3]

Sample Output for n=5

[11111, 1112, 1121, 113, 1211, 122, 131, 2111, 212, 221, 23, 311, 32]

```
*/
```

Code : 4

```
package Topic_09_RecursionWithArrayList;
```

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
```

```
public class D_GetMazePaths {
    public static void main(String[] args) throws Exception {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        int m = s.nextInt();
        System.out.println(getMazePaths(1, 1, n, m));
    }

    public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
        if (sr == dr && sc == dc) {
            ArrayList<String> res = new ArrayList<String>();
            res.add("");
            return res;
        }

        ArrayList<String> hPaths = new ArrayList<String>();
        ArrayList<String> vPaths = new ArrayList<String>();

        if (sc < dc)
            hPaths = getMazePaths(sr, sc + 1, dr, dc);//2

        if (sr < dr)
            vPaths = getMazePaths(sr + 1, sc, dr, dc);//1

        ArrayList<String> paths = new ArrayList<String>();

        for (String s : hPaths) {
            paths.add("h" + s);
        }
        for (String s : vPaths) {
            paths.add("v" + s);
        }
        return paths;
    }
}

/*
Sample Output for 3x3
[hhvv, hvhv, hvvh, vhhv, vvhv, vvhh]

*/
```

Code : 5

```
package Topic_09_RecursionWithArrayList;
```

```
import java.util.ArrayList;
```

```
import java.util.Scanner;
```

```
public class E_GetMazePathsWithJumps {
```

```
    public static void main(String[] args) throws Exception {
```

```
        Scanner s = new Scanner(System.in);
```

```
        int n = s.nextInt();
```

```
        int m = s.nextInt();
```

```
        System.out.println(getMazePathsWithJumps(1, 1, n, m));
```

```
    }
```

```
    public static ArrayList<String> getMazePathsWithJumps(int sr, int sc, int dr, int dc) {
```

```
        if (sr == dr && sc == dc) {
```

```
            ArrayList<String> res = new ArrayList<String>();
```

```
            res.add("");
```

```
            return res;
```

```
        }
```

```
        ArrayList<String> paths = new ArrayList<String>();
```

```
        //Horizontal moves
```

```
        for (int ms = 1; ms <= dc - sc; ms++) {
```

```
            ArrayList<String> hPaths = getMazePathsWithJumps(sr, sc + ms, dr, dc);
```

```
            for (String s : hPaths) {
```

```
                paths.add("h" + ms + s);
```

```
            }
```

```
        }
```

```
        //vertical moves
```

```
        for (int ms = 1; ms <= dr - sr; ms++) {
```

```
            ArrayList<String> vPaths = getMazePathsWithJumps(sr + ms, sc, dr, dc);
```

```
            for (String s : vPaths) {
```

```
                paths.add("v" + ms + s);
```

```
            }
```

```
        }
```

```
        //diagonal moves
```

```
        for (int ms = 1; ms <= dr - sr && ms <= dc - sc; ms++) {
```

```
            ArrayList<String> dPaths = getMazePathsWithJumps(sr + ms, sc + ms, dr, dc);
```

```
            for (String s : dPaths) {
```

```
                paths.add("d" + ms + s);
```

```
            }
```

```
        }
```

```
        return paths;
```

```
    }
```

```
}
```

```
/*
```

```
Sample Output 2x2
```

```
[h1v1, v1h1, d1]
```

```
Sample Output 3x3
```

[h1h1v1v1, h1h1v2, h1v1h1v1, h1v1v1h1, h1v1d1, h1v2h1, h1d1v1, h2v1v1, h2v2, v1h1h1v1, v1h1v1h1, v1h1d1, v1h2v1, v1v1h1h1, v1v1h2, v1d1h1, v2h1h1, v2h2, d1h1v1, d1v1h1, d1d1, d2]

\*/

