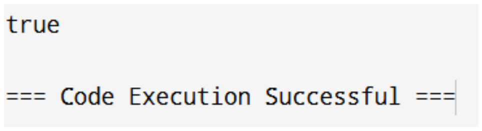
1. Anagram program:

CODE:

```
import java.util.*;
class HelloWorld {
  public static boolean anagram(String s1,String s2){
    if(s1.length()!=s2.length()) return false;
    int[] f=new int[26];
    for(int i=0;i<s1.length();i++){
      f[s1.charAt(i)-'a']++;
    }
    for(int i=0;i<s2.length();i++){
      f[s2.charAt(i)-'a']--;
    }
    for(int i=0;i<26;i++){
      if(f[i]!=0) return false;
    }
    return true;
  }
  public static void main(String[] args) {
    String s1="geeks";
    String s2="kseeg";
    System.out.println(anagram(s1,s2));
  }
}
```

OUTPUT:



```
true

=== Code Execution Successful ===
```

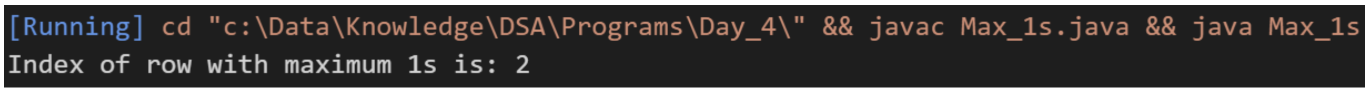
Time Complexity: O(n)

2. Row with max 1's:

CODE:

```
public class Max_1s{
  static final int R=4;
  static final int C=4;
  public static int rowwithmax1s(int[][] mat){
    int maxrow=-1, r=0, c=C-1;
    while(r<R && c>=0){
      if(mat[r][c]==0){
        r++;
      }
      else{
        maxrow=r;
        c--;
      }
    }
    return maxrow;
  }
  public static void main(String[] args){
        int[][] mat={{0,0,0,1},{0,1,1,1},{1,1,1,1},{0,0,0,0}};
        System.out.println("Index of row with maximum 1s is: "+rowwithmax1s(mat));
    }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_4\" && javac Max_1s.java && java Max_1s
Index of row with maximum 1s is: 2
```

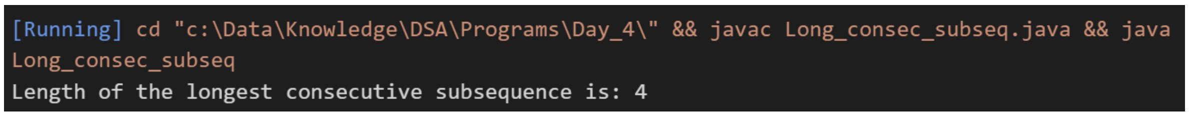
Time Complexity: O(m+n)

3. Longest consecutive subsequence:

CODE:

```
import java.util.*;
import java.io.*;
class Long_consec_subseq{
    static int longconssub(int a[], int n){
        HashSet<Integer> s=new HashSet<Integer>();
        int r=0;
        for(int i=0;i<n;i++) s.add(a[i]);
        for(int i=0;i<n;i++){
            if(!s.contains(a[i]-1)){
                int j=a[i];
                while(s.contains(j)) j++;
                if(r<j-a[i]) r=j-a[i];
            }
        }
        return r;
    }
    public static void main(String[] args){
        int a[]={1,9,3,10,4,20,2};
        int n=a.length;
        System.out.println("Length of the longest consecutive subsequence is: "+longconssub(a,n));
    }
}
```

OUTPUT:



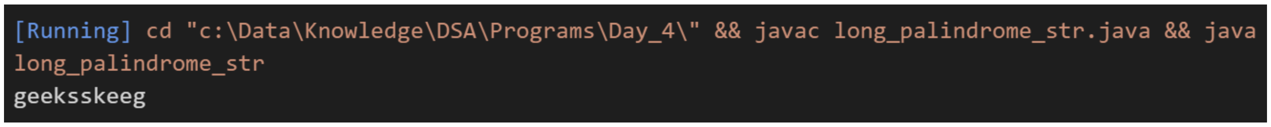
```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_4\" && javac Long_consec_subseq.java && java
Long_consec_subseq
Length of the longest consecutive subsequence is: 4
```

Time Complexity: O(n)

4. Longest palindrome in a string:

CODE:

```
public class long_palindrome_str{
    static String longpalstr(String s){
        int n=s.length();
        if(n==0) return "";
        int st=0, maxlen=1;
        for(int i=0;i<n;i++){
            for(int j=0;j<=1;j++){
                int l=i;
        int h=i+j;
        while(l>=0 && h<n && s.charAt(l)==s.charAt(h)){
          int currlen=h-l+1;
          if(currlen>maxlen){
            st=l;
            maxlen=currlen;
          }
          l--;
          h++;
        }
      }
    }
    return s.substring(st,st+maxlen);
  }
  public static void main(String[] args){
    String s="forgeeksskeegfor";
    System.out.println(longpalstr(s));
  }
}
```
OUTPUT:



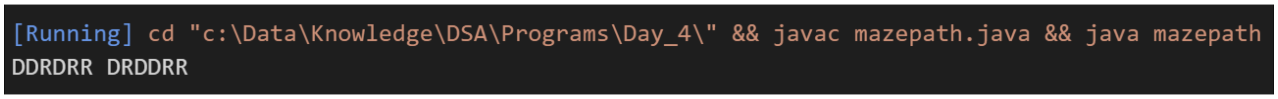
```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_4\" && javac long_palindrome_str.java && java
long_palindrome_str
geeksskeeg
```
Time Complexity: O(n^2)

5. Rat in a maze problem:

CODE:
```
import java.util.*;
public class mazepath{
  static String direction = "DLRU";
  static int[] dr={1,0,0,-1};
  static int[] dc={0,-1,1,0};
  static boolean valid(int r, int c, int n, int[][] maze){
    return r>=0 && c>=0 && r<n && c<n && maze[r][c]==1;
  }
  static void findpath(int r, int c, int[][] maze, int n, ArrayList<String> a, StringBuilder currpath){
    if(r==n-1 && c==n-1){
      a.add(currpath.toString());
      return;
    }
    maze[r][c]=0;
    for(int i=0;i<4;i++){
      int nextr=r+dr[i];
      int nextc=c+dc[i];
      if(valid(nextr, nextc, n, maze)){
        currpath.append(direction.charAt(i));
        findpath(nextr, nextc, maze, n, a, currpath);
        currpath.deleteCharAt(currpath.length()-1);
      }
```

```
        }
        maze[r][c]=1;
    }
    public static void main(String[] args){
        int[][] maze={{1,0,0,0},{1,1,0,1},{1,1,0,0},{0,1,1,1}};
        int n=maze.length;
        ArrayList<String> res=new ArrayList<>();
        StringBuilder currpath=new StringBuilder();
        if(maze[0][0]!=0 && maze[n-1][n-1]!=0){
            findpath(0,0,maze,n,res,currpath);
        }
        if(res.size()==0) System.out.println(-1);
        else
            for(String p:res)
                System.out.print(p+" ");
        System.out.println();
    }
}
```

OUTPUT:



```
[Running] cd "c:\Data\Knowledge\DSA\Programs\Day_4\" && javac mazepath.java && java mazepath
DDRDRR DRDDRR
```

Time Complexity: O(3^(m*n))