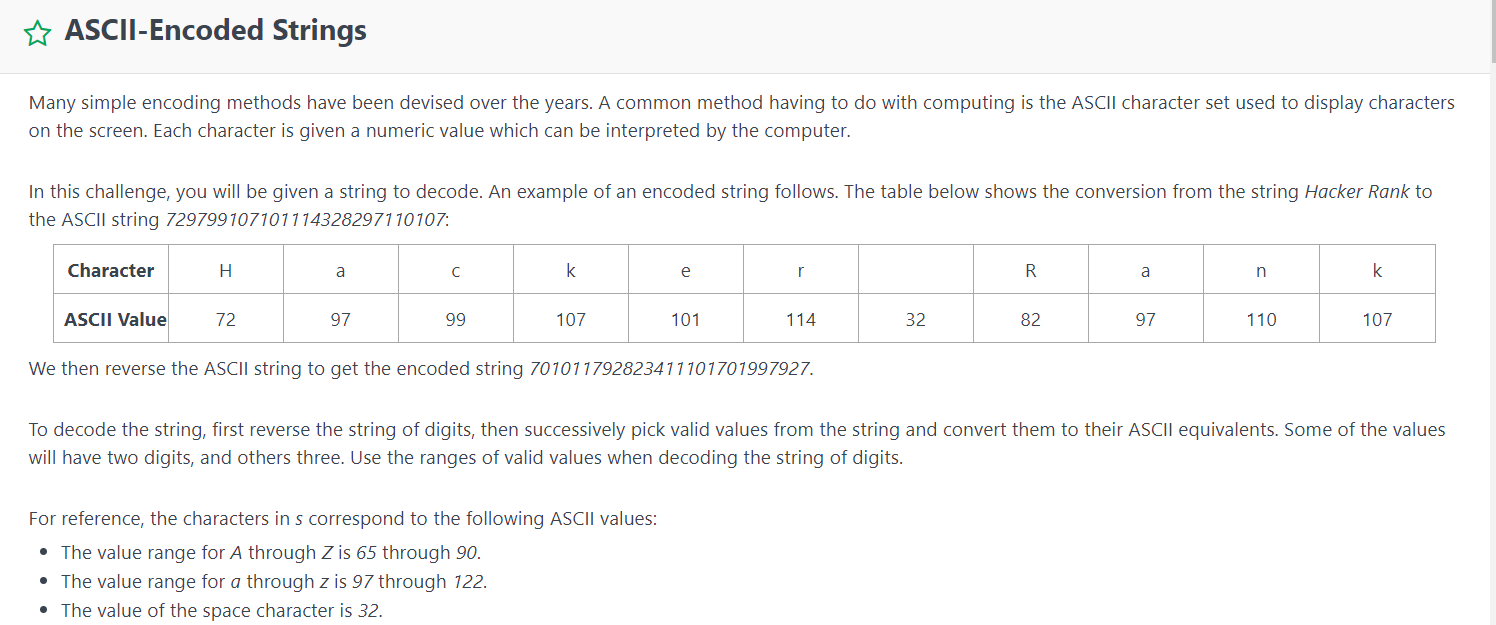
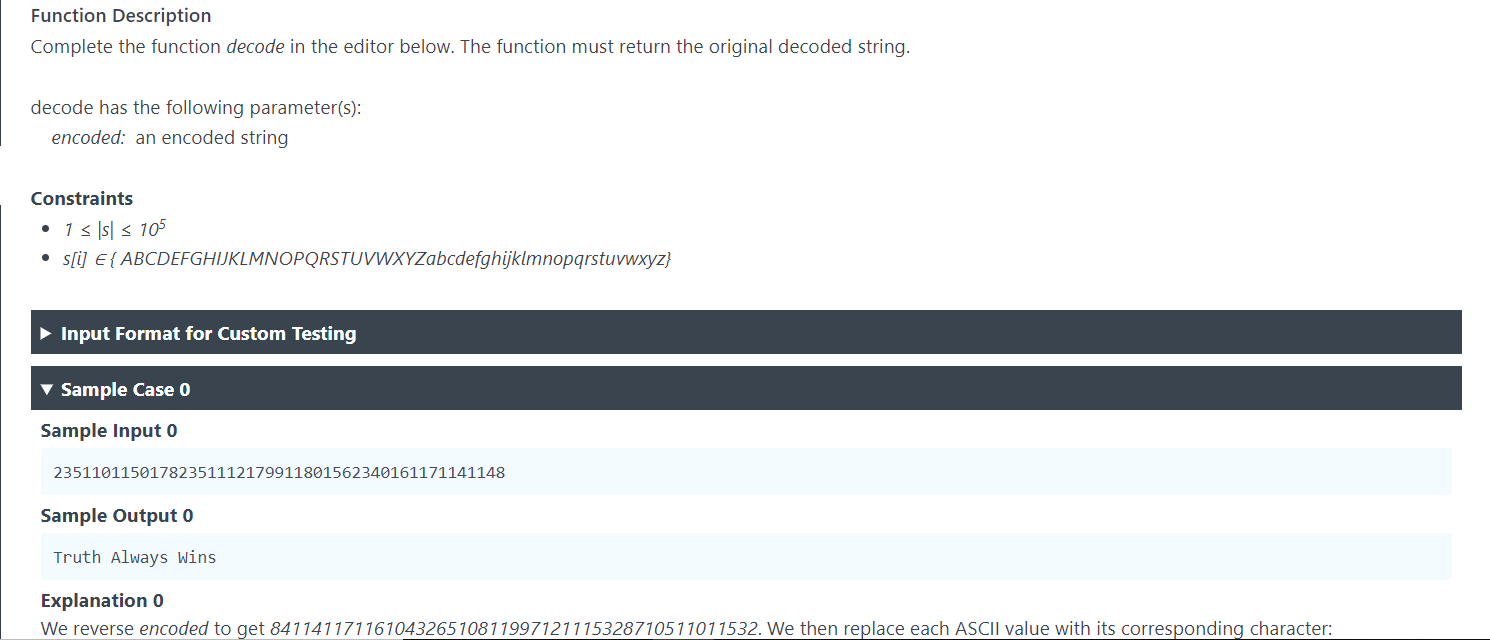
Twitter test:

1.





Solution:

// Complete the decode function below.

static String decode(String encoded) {

StringBuilder encodedSB=new StringBuilder();

StringBuilder result=new StringBuilder();

encodedSB.append(encoded);

encodedSB=encodedSB.reverse();

//System.out.println(Integer.parseInt(encoded)+5);

if(encodedSB.length()<3){

// System.out.println(Integer.parseInt(encoded)+5);

if(Integer.parseInt(encoded)!=32&&Integer.parseInt(encoded)<65||(Integer.parseInt(encoded)>90&&Integer.parseInt(encoded)<97)||Integer.parseInt(encoded)>122){

System.out.println(Integer.parseInt(encoded)+5);

return "";

}

else

return Character.toString((char)Integer.parseInt(encodedSB.toString()));

}

else{

int i=0,j=2;

int character=0;

while((i+j)<=encodedSB.length()){

character=Integer.parseInt(encodedSB.substring(i,i+j));

if(character>64&&character<91||character>96&&character<100||character==32){

result.append((char)character);

}

else if(character>9&&character<13){

if(i+j!=encodedSB.length()){

character=Integer.parseInt(String.valueOf(character)+Character.toString(encodedSB.charAt(i+j)));

if(character>99&&character<123)

result.append((char)character);

j=j+1;

}

else return "";

}

else

return "";

i=i+j;

j=2;

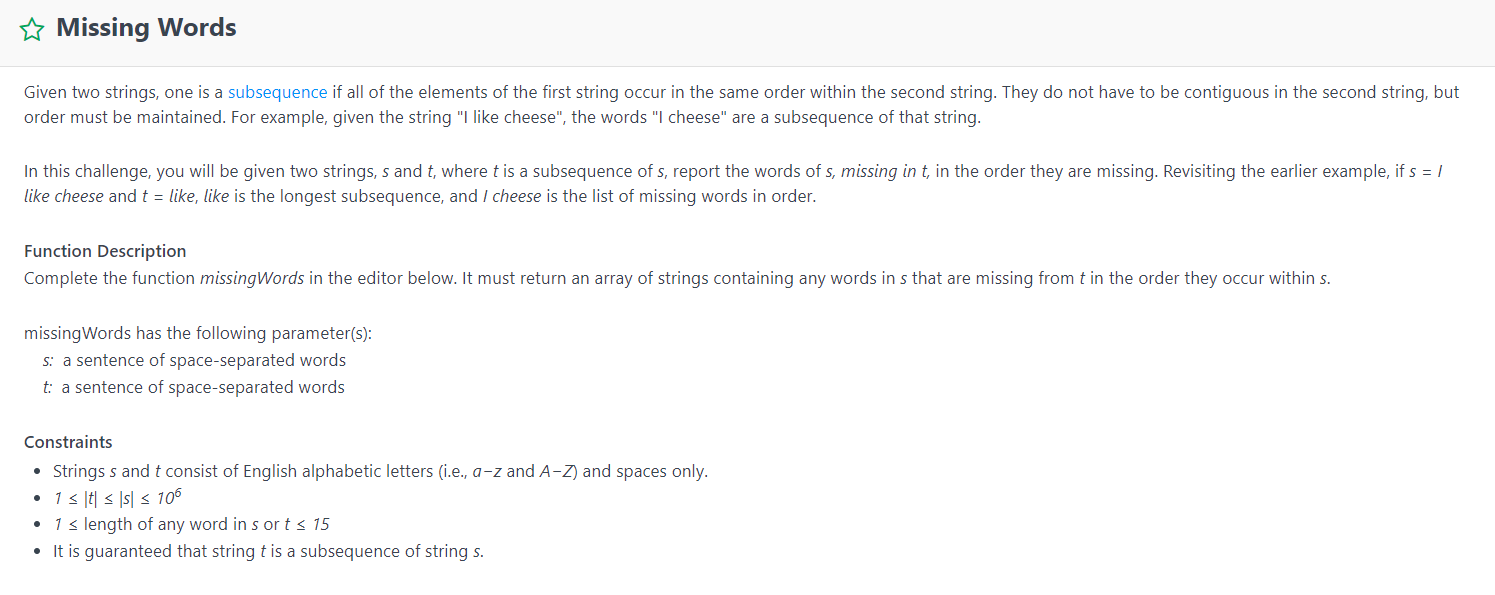
}

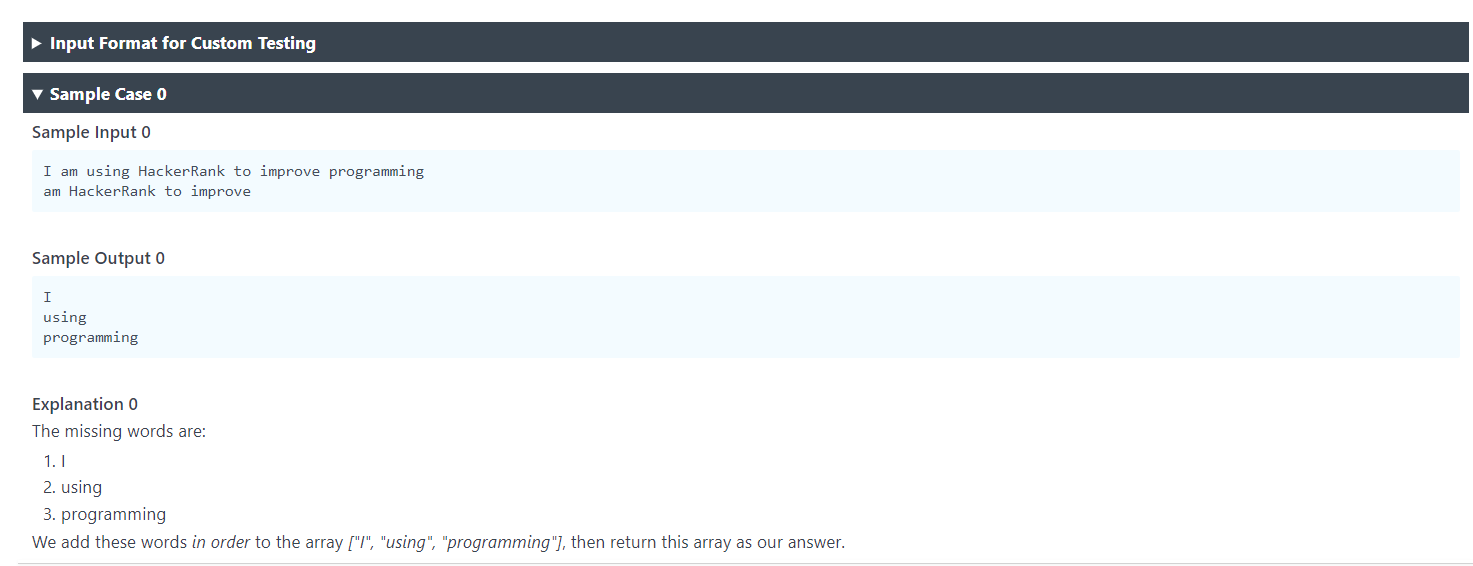
return result.toString();

}

}

2.





Solution:

class Result {

/\*

\* Complete the 'missingWords' function below.

\*

\* The function is expected to return a STRING\_ARRAY.

\* The function accepts following parameters:

\* 1. STRING s

\* 2. STRING t

\*/

public static List<String> missingWords(String s, String t) {

// Write your code here

//Set<String> all\_words=new

int from\_index=0, index=0,i=0;

List<String> result= new ArrayList<String>();

String[] s\_words=s.split(" ");

List<String> s\_words\_list=Arrays.asList(s\_words);

int s\_words\_size=s\_words\_list.size();

if(t==null){

return s\_words\_list;

}

else{

String[] t\_words= t.split(" ");

List<String> t\_words\_list=Arrays.asList(t\_words);

int t\_words\_size=t\_words\_list.size();

while(i<t\_words.length){

if(s\_words\_list.subList(from\_index,s\_words\_size).contains(t\_words\_list.get(i))){

index=s\_words\_list.subList(from\_index,s\_words\_size).indexOf(t\_words\_list.get(i));

result.addAll(result.size(),s\_words\_list.subList(from\_index,from\_index+index));

from\_index=from\_index+index+1;

i++;

}

else

return null;

}

result.addAll(result.size(),s\_words\_list.subList(from\_index,s\_words\_size));

return result;

}

}

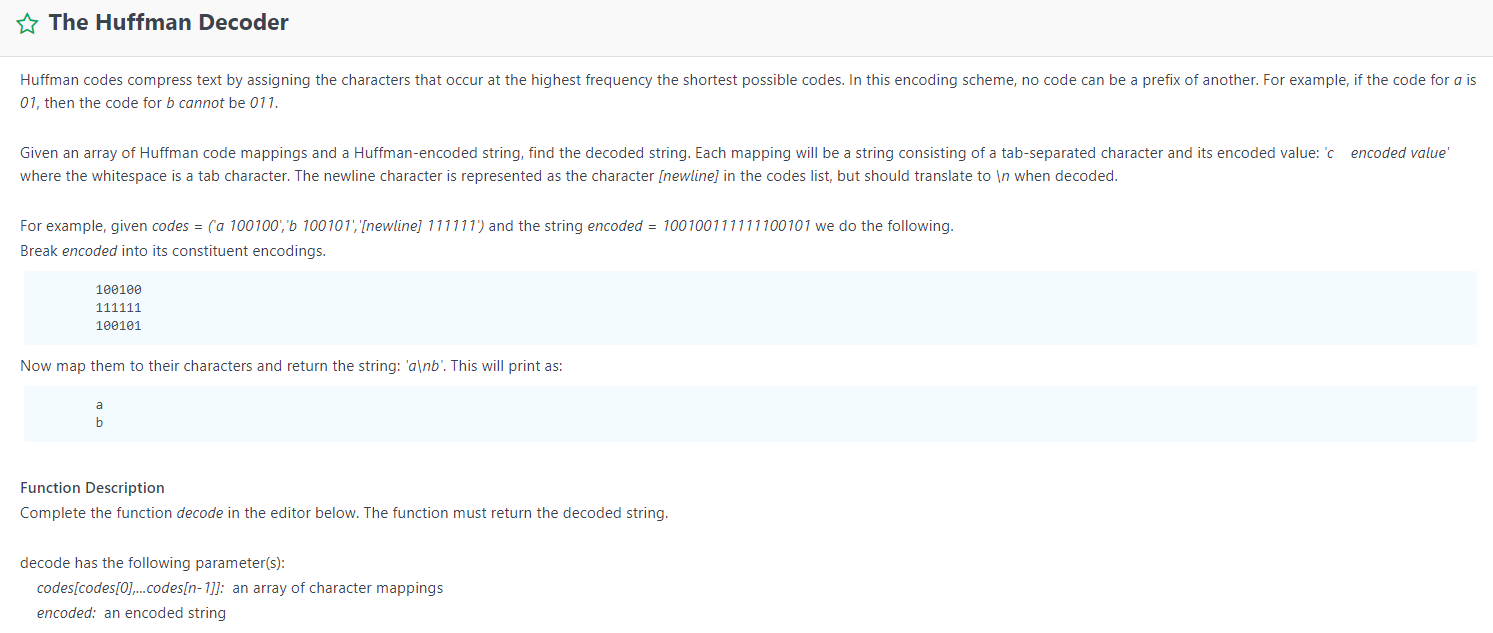
}

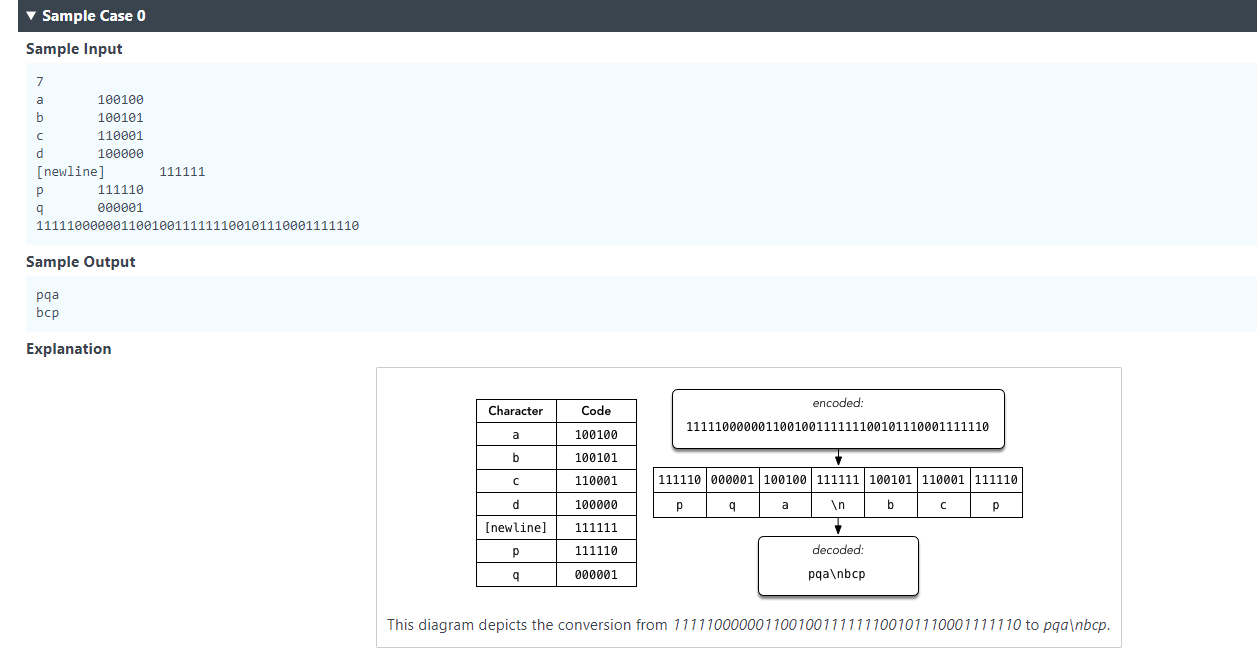
//System.out.println(t\_words\_list.get(i)+" found at index: "+s\_words\_list.subList(index,s\_words\_size).indexOf(t\_words\_list.get(i))+" in the subList: "+ s\_words\_list.subList(index,s\_words\_size));

//System.out.println("result from & to: previous: "+previous\_index+" index: "+((index+previous\_index)-1));

//System.out.println(result);

3.





Solution:

// Complete the decode function below.

static String decode(List<String> codes, String encoded) {

int start\_index=0;

List<String> code\_value= new ArrayList<>();

List<String> letters= new ArrayList<>();

StringBuilder result=new StringBuilder();

for(int j=0;j<codes.size();j++){

String[] w=codes.get(j).split("\t");

code\_value.add(w[1]);

letters.add(w[0]);

}

for(int i=1;i<=encoded.length();i++){

if(code\_value.contains(encoded.substring(start\_index,i))){

if(!letters.get(code\_value.indexOf(encoded.substring(start\_index,i))).equals("[newline]"))

result.append(letters.get(code\_value.indexOf(encoded.substring(start\_index,i))));

else

result.append("\n");

start\_index=i;

}

}

return result.toString();

}