Q1.

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

public class KnapsackProblem {

   private static List<List<Integer>> knapsack(List<List<Integer>> result,int[] s,int[] w, int[]
v, int maxW, int minV, List<Integer> subset, int currIndex){

      if(maxW >= 0 && minV <= 0){
         result.add(new ArrayList<>(subset));
         // do not return, because we can still find the other combinations
      }

      if(currIndex == s.length){
         return result;
      }

      knapsack(result,s,w,v,maxW, minV, subset, currIndex +1);
      subset.add(s[currIndex]);
      knapsack(result,s,w,v,maxW-w[currIndex], minV-v[currIndex], subset, currIndex +1);
      subset.removeLast();
      return result;
   }
   public static List<List<Integer>> solve(int[] s,int[] w, int[] v, int maxW, int minV){
      return knapsack(new ArrayList<>(),s,w,v,maxW, minV,new ArrayList<>(),0);
   }

   public static void main(String[] args){
      System.out.println(solve(new int[]{1,2,3},new int[]{4,8,9},new int[]{2,4,6},14,5));
   }
}
```

Q2.

```
class Solution {

public static List<List<Integer>> permute(List<List<Integer>> result, List<Integer>
permutation, int[] nums, int index){
if(index == nums.length){
result.add(new ArrayList(permutation));
return result;
}

for(int n: nums){
if(permutation.contains(n)) continue;
permutation.add(n);
permute(result,permutation,nums,index+1);
permutation.removeLast();
}

return result;
}
public static List<List<Integer>> permute(int[] nums) {
return permute(new ArrayList<>(), new ArrayList<>(), nums,0);
}
}
```

Q3.

```
class Solution {
static String[] dict = {"","","abc","def","ghi","jkl","mno","pqrs","tuv","wxyz"};

private static List<String> letterCombinations(List<String> result, StringBuilder temp,
String digits){
if(digits.isEmpty()){
result.add(temp.toString());
return result;
}
String keypadValues = dict[Integer.parseInt(String.valueOf(digits.charAt(0)))];
for(char c: keypadValues.toCharArray()){
temp.append(c);
letterCombinations(result,temp,digits.substring(1));
temp.setLength(temp.length()-1);
}
return result;
}


public static List<String> letterCombinations(String digits) {
if(digits.isBlank()) return new ArrayList<>();
return letterCombinations(new ArrayList<String>(), new StringBuilder(), digits);
}

}
```