Homework 2: Array problem II

Problem 1:

How to pack as much value with a weight constraint W? There are four items you can put into your pack.

The following table lists the value and the corresponding weight for each item:

Item i	Value v_i	Weight w_i
1	15	6
2	12	4
3	9	5
4	7	3
5	4	2

Recursion formula:

$$f(k, g) = \begin{cases} f(k-1,g) & \text{if } w_k > g \quad (\text{kth item放不下去}) \\ \max\{ v_k + f(k-1, g - w_k),\ f(k-1,g) \} & \text{if } w_k \leq g \text{ and } k>0 \end{cases}$$

Where $f(0, g)=0$, $f(k, 0)=0$

k: ***number of item****, g:* ***current weigt***

Please write a program to select the items with maximum value under the constraint of weight W provided by the user.

Please let the user input W and stop when inputting CTRL+D.

Please print:

1. The values of all cases of *f(k, g) where* $1 \leq k \leq 5$, $1 \leq g \leq W$
2. The maximum value of *f(k, g)* and list the selected items.

Input/Output Example:

```
14
  0   0   0   0   0  15  15  15  15  15  15  15  15  15
  0   0   0  12  12  15  15  15  15  27  27  27  27  27
  0   0   0  12  12  15  15  15  21  27  27  27  27  27
  0   0   7  12  12  15  19  19  22  27  27  28  34  34
  0   4   7  12  12  16  19  19  23  27  27  31  34  34
 34
  4   2   1
7
  0   0   0   0   0  15  15
  0   0   0  12  12  15  15
  0   0   0  12  12  15  15
  0   0   7  12  12  15  19
  0   4   7  12  12  16  19
 19
  4   2
```

All output format: %3d

Problem 2: Use “vector” only

Given a positive integer *n*, print out the position of all 1’s in its binary representation. The position of the least significant bit is 0.

Example:

The positions of 1’s in the binary representation of 13 are 0, 2,3.

a. Let the user continuously input n and stop when inputting CTRL+D.

b. Please use 'vector' only.

c. You should use at least three vector functions:

example: clear() ,pushback(),size(),…

Input/Output Example:

```
1024
10
1023
0 1 2 3 4 5 6 7 8 9
1035
0 1 3 10
```

Each output data is separated by a blank(space)

Problem 3:

Write a program that reads in a set of positive integers, representing test scores for a class, and outputs how many times a particular number appears in the list. You may assume that the data set has at most 100 numbers and -999 marks the end of the input data. The numbers must be output in increasing order. Please let the user input scores and stop the program when inputting CTRL+D.

Input/Output Example:

```
Input scores:-999
Input scores:80 60 90 90 80 65 97 100 80 65 90 65 -999
 60: 1
 65: 3
 80: 3
 90: 3
 97: 1
100: 1
Input scores:80 80 75 90 92 75 -999
 75: 2
 80: 2
 90: 1
 92: 1
Input scores:
```

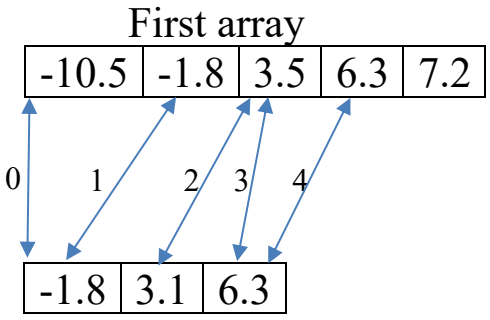
Problem 4:
Write a function merge() that merges the contents of two sorted arrays and return the result array(sorted). The merge() has the following prototype:

vector<double> merge(vector<double> &array1, vector<double> &array2)

The function should not assume that both its input parameter arrays are the same length but can assume that one array does not contain two copies of the same value. Result array should also contain no duplicate values. Let the user input the lengths of arrays first. Please use **vector** to complete this problem.

Then let the user input the target and output the target's index in the merged-array. If the target is not in the array, return -1. Please let the user input target, and stop when inputting CTRL+D.

Example:



Result array

-10.5	-1.8	3.1	3.5	6.3	7.2

When one of the input arrays has been exhausted, do not forget to copy the remaining data in the other array into the result array.
Remember that the arrays input to this function must already be sorted.

Hint1:

```
vector<double> merge(vector<double> &array1, vector<double> &array2){
    vector<double> v;
    int j=0,k=0;
    while (j<array1.size() || k<array2.size()){
        if( j == array1.size()){
            v.push_back(array2[k]);
            k++;
        }
        else if( k==array2.size()){
            …
        }
        else if( array1[j]>array2[k] ){
            …
        }
        else if( array1[j]==array2[k] ){
            …
        }
        else{
            …
        }
    }
    return v;
}
```

Hint2:

```
void vsort(vector<double> &v){
    …
}
```

Hint3:

```
int binarySearch(vector<double> &v, double tar){
    …       return -1;
}
```

Input/Output Example: