

Suma de Combinaciones

Dado un arreglo de enteros **A** y una suma **B**, encuentra todas las únicas combinaciones en **A** donde la suma es igual a **B**. El mismo número repetido puede ser escogido de **A**, ilimitadas veces.

Nota: Todos los números son enteros, los elementos en la combinación (a_1, a_2, \dots, a_k) deben estar en orden ascendente $(a_1 \leq a_2 \leq a_3, \dots, a_k)$. Así mismo las combinaciones deben ser ordenadas en orden creciente. Si no hay combinación posible la respuesta será "Empty" (Vacio, sin comillas).

Ejemplo:

Dado **A** = {2, 4, 6, 8} y **B** = 8, un conjunto de soluciones es: {2, 2, 2, 2} {2, 2, 4} {2, 6} {4, 4} {8}

Formato de archivo de entrada

La primera línea contiene un entero **T** es el número de casos. Cada caso tiene tres líneas.

La primera línea es **N**, tamaño del arreglo **A**. La segunda línea contiene **N** enteros, $\{a_1, a_2, a_3, \dots, a_n\}$ La tercera línea contiene el número **B**.

Restricciones

1. $1 \leq T \leq 500$
2. $1 \leq N \leq 12$
3. $1 \leq a_i \leq 9$
4. $1 \leq B \leq 30$

Formato de salida

La salida es una línea por cada caso, cada subconjunto encerrado en () y cada conjunto de enteros deberá ser separado por espacios. (Vea los ejemplos)

Entrada Simple 0

```
3
4
7 2 6 5
16
11
6 5 7 1 8 2 9 9 7 7 9
6
4
```

5 2 2 6
3

Salida para la Entrada Simple 0

(2 2 2 2 2 2 2 2)(2 2 2 2 2 6)(2 2 2 5 5)(2 2 5 7)(2 2 6 6)(2 7 7)(5 5 6)
(1 1 1 1 1 1)(1 1 1 1 2)(1 1 2 2)(1 5)(2 2 2)(6)

Empty

```
// Solution Combination Sum
import java.io.*;
import java.util.*;

public class GFG{

    public static void main(String[]arg) throws IOException{
        new GFG().read();
    }

    public void read() throws IOException{
        BufferedReader in = new BufferedReader(new
            InputStreamReader(System.in));
        StringTokenizer st;
        int T,N,i,A[],B;
        T = Integer.parseInt(in.readLine());
        while(T-->0){
            N = Integer.parseInt(in.readLine());
            A = new int[N];
            st = new StringTokenizer(in.readLine());
            for(i = 0; i < N; i++) A[i] =
                Integer.parseInt(st.nextToken());
            B = Integer.parseInt(in.readLine());
            System.out.println(getCombinations(A,B));

        }
    }

    public String getCombinations(int[]A, int B){
        String set = "";
        mergeSort(A,1,A.length);
        return set;
    }

    void mergeSort(int[]A, int p, int r){
        int q;
        if(p < r){
            q = (p+r)/2;
            mergeSort(A,p,q);
            mergeSort(A,q+1,r);
            merge(A,p,q,r);
        }
    }
}
```

```

void merge(int[] A, int p, int q, int r){
    int n1 = q - p + 1, n2 = r - q, L[], R[];
    L=new int[n1+1];
    R=new int[n2+1];
    L[n1]=R[n2]=Integer.MAX_VALUE;
    fillArray(L,A,1,n1,p-1);
    fillArray(R,A,1,n2,q);
    toMerge(1,1,p,r,A,L,R);
}

void fillArray(int[] array, int[] A, int i, int n, int j){
    if(i <= n){
        array[i-1] = A[i+j-1];
        fillArray(array,A,i+1,n,j);
    }
}

void toMerge(int i, int j, int p, int r, int[] A, int[] L, int[] R){
    if(p <= r){
        if(L[i-1]<=R[j-1]){
            A[p-1]=L[i-1];
            toMerge(i+1,j,p+1,r,A,L,R);
        }else{
            A[p-1]=R[j-1];
            toMerge(i,j+1,p+1,r,A,L,R);
        }
    }
}
}

```

La entrega del trabajo será hasta el día 30 de Septiembre, horas 23:00. Solo es necesario enviar los archivos.java en una carpeta comprimida de esta forma:

ApellidoPaterno _ ApellidoPaterno _ Nombre _ Grupo.zip

Por ejemplo:

Perez _ Zabalaga _ Fernando _ Grupo _ 2.zip

Esto deberá ser enviado a al e-mail:

mastercatowo@gmail.com