```

/*
Si scriva un metodo costruisciVettore che riceve in ingresso due vettori di
interi v1, v2 e
restituisce in output un vettore di interi v3 contenente gli elementi x di
v1 che non occorrono esattamente x volte in v2.
Ad esempio, nel caso in cui v1= [2, 5, 15, 1, 2], v2= [ 5, 2, 5, 1, 5, 1,
5, 2, 5],
il metodo restituisce v3 = [15, 1].
*/

```

```

*/

```

```

public class Esercizio2B{
    public static void main(String[] args){

        int[] v1= {2, 5, 15, 1, 2};
        int[] v2= {5, 2, 5, 1, 5, 1, 5, 2, 5};
        int[] c=costruisciVettore(v1,v2);
        for (int i=0;i<c.length;i++){
            System.out.print(c[i]+" ");
        }

        public static int[] costruisciVettore(int[] v1,int[] v2){
            int dim= calcolaDim(v1,v2);
            int[] R=new int[dim];
            int pos=0;
            for (int i=0;i<v1.length;i++)
                if (presente(v2,v1[i])!= v1[i]) {
                    R[pos]=v1[i];
                    pos++;
                }

            return R;
        }

        public static int calcolaDim(int[] v1,int[] v2){
            int dim= 0;
            for (int i=0;i<v1.length;i++)
                if (presente(v2,v1[i])!= v1[i])
                    dim++;

            return dim;
        }

        public static int presente(int[] v,int val){
            int cont= 0;
            for (int i=0;i<v.length;i++)
                if (v[i]== val)
                    cont++;

            return cont;
        }

    }
}

```

```

/*
Si realizzi una classe Matrice per rappresentare matrici di interi che
contenga almeno i seguenti metodi:
1. un metodo verifica che riceve in ingresso una matrice quadrata M di
interi di dimensione dispari e restituisce in output un booleano.
In particolare il metodo restituisce true se per ogni elemento presente
sulla colonna centrale di M esiste un suo divisore intero
sulla riga centrale di M. Il metodo restituisce false altrimenti (si
veda esempio).
2. un metodo controlla che riceve in ingresso una matrice quadrata di
interi M di dimensione n ed un array di interi V di dimensione n,
e restituisce in output un booleano. In particolare il metodo

```

- restituisce true se ogni elemento $V[i]$ è uguale alla somma degli elementi della riga i di M oppure è uguale al prodotto degli elementi della colonna i di M . Il metodo restituisce false altrimenti (si veda esempio).
3. un metodo `creaMatrice` che riceve in ingresso una matrice M di interi e restituisce in output una matrice di interi contenente tutte le colonne di M di indice dispari contenenti solo elementi pari (si veda esempio).
4. un metodo `main` che legge una matrice M di numeri interi e provvede, invocando opportunamente i metodi proposti, a realizzare i compiti di cui ai punti (1), (2) e (3) .

```
*/
```

```
public class Esercizio3B{

    public static boolean verifica(int[] [] M)
    {
        for (int i = 0; i < M.length; i++)
        {
            boolean esistedivisore = false;
            for (int j = 0; j < M.length; j++)
                if (M[i][M.length/2] %M[M.length/2]
                    [j]== 0)
                    esistedivisore = true;
            if (! esistedivisore)
                return false;
        }
        return true;
    }

    public static boolean controlla(int[] [] M, int [] V)
    {
        for (int i = 0; i < V.length; i++)
            if ((V[i] != SommaElementi(M, i)) && (V[i] !=
                ProdottoColonna(M,i)))
                return false;
        return true;
    }

    public static int SommaElementi(int[] [] M, int r)
    {
        int somma=0;
        for (int j = 0; j < M[0].length; j++)
            somma += M[r][j];

        return somma;
    }

    public static int ProdottoColonna(int[] [] M, int c)
    {
        int prodotto=1;
        for (int i = 0; i < M.length; i++)
            prodotto *= M[i][c];
        return prodotto;
    }

    public static int[] [] creaMatrice(int[] [] M)
    {
        int NumC = CalcolaColonne(M);
        int [] [] SM = new int[M.length] [NumC];
        int col=0;
        for (int j = 1; j < M.length && ColonnaOK(M,j); j+=2){
            for (int i = 0; i < M.length; i++)
                SM[i][col] = M[i][j];

            col++;
        }
        return SM;
    }
}
```

```

public static boolean ColonnaOK(int[] [] M, int col)
{
    for (int i = 0; i < M.length; i++)
        if (M[i][col] %2 !=0)
            return false;
    return true;
}

public static int CalcolaColonne(int[] [] M)
{
    int NumC =0;
    for (int j = 1; j < M.length; j+=2)
        if (ColonnaOK(M,j))
            NumC++;

    return NumC;
}

public static void stampa(int vettore[])
{
    for (int i = 0; i < vettore.length; i++)
    {
        System.out.print(vettore[i] + "\t");
    }
    System.out.println();
}

public static void stampa(int matrice[] [])
{
    for (int i = 0; i < matrice.length; i++)
    {
        for (int j = 0; j < matrice[0].length; j++)
            System.out.print(matrice[i][j] +
                "\t");
        System.out.println();
    }
}

public static void main(String args[])
{
    int[] [] M = {{1,8, 10, 2, 2}, {2, 2, 14, 4,
1}, {3, 2,6, 12, 7}, {1,4,24,2,1}, {3, 2, 9, 2, 2}};
    int [] V={18,23,30,32,28};
    stampa(M); stampa(V);
    System.out.println("Risultato di verifica(M):");
    System.out.println(verifica(M));
    System.out.println("Risultato di controlla(M,V):");
    System.out.println(controlla(M,V));
    System.out.println("Risultato di creaMatrice(M):");
    int [] [] SM = creaMatrice(M);
    stampa(SM);
}
}

```