/*11. Design and implement in Java to find a subset of a given set S = {Sl, S2,.....,Sn} of n positive integers whose SUM is equal to a given positive integer d. For example, if S = {1, 2, 5, 6, 8} and d=9, there are two solutions {1,2,6}and {1,8}. Display a suitable message, if the given problem instance doesn't have a solution*/

```
import java.util.Scanner;

public class SumOfsubset
{
	final static int MAX = 10;
	static int n;
	static int S[];
	static int soln[];
	static int d;

	public static void main(String args[])
	{
		S = new int[MAX];
		soln = new int[MAX];
		int sum = 0;
		Scanner scanner = new Scanner(System.in);
		System.out.println("Enter number of elements: ");
		n = scanner.nextInt();

		System.out.println("Enter the set in increasing order: ");
		for (int i = 1; i <= n; i++)
			S[i] = scanner.nextInt();
		System.out.println("Enter the max. subset value(d): ");
		d = scanner.nextInt();
		for (int i = 1; i <= n; i++)
			sum = sum + S[i];
		if (sum < d || S[1] > d)
			System.out.println("No Subset possible");
		else
			SumofSub(0, 0, sum);
		scanner.close();
	}

	static void SumofSub(int i, int weight, int total)
	{
		if (promising(i, weight, total) == true)
			if (weight == d)
			{
				for (int j = 1; j <= i; j++)
				{
					if (soln[j] == 1)
						System.out.print(S[j] + "  ");
				}
```

```
                    System.out.println();
                }
                else
                {
                    soln[i + 1] = 1;
                    SumofSub(i + 1, weight + S[i + 1], total - S[i + 1]);
                    soln[i + 1] = 0;
                    SumofSub(i + 1, weight, total - S[i + 1]);
                }
        }

        static boolean promising(int i, int weight, int total)
        {
            return ((weight + total >= d) && (weight == d || weight + S[i + 1] <= d));
        }
}
```