**package** Abstract.Dynamic\_Programming;

**class** rodCutRecursion

{

**public int**[] **pieceLength**;

**public int**[] **pieceProfit**;

**public int totalLength**;

**public int maxProfit**;

**public int**[] **memorization**;

**public int temp**;

**public** rodCutRecursion(**int**[] pieceLength, **int**[] pieceProfit, **int** totalLength) {

**this**.**pieceLength** = pieceLength;

**this**.**pieceProfit** = pieceProfit;

**this**.**totalLength** = totalLength;

**this**.**memorization** = **new int**[**this**.**pieceLength**.**length**];

}

**public int** findMaxProfit(**int** totalLength)

{

**if** (totalLength == 0) **return** 0;

**if** (totalLength<0) **return** -9999;

**maxProfit** = -999999;

**for** (**int** i=0;i<**pieceLength**.**length**;i++)

{

**maxProfit** = Math.*max*(**maxProfit**, (**pieceProfit**[i] + findMaxProfit(totalLength - **pieceLength**[i])));

}

**return maxProfit**;

}

}

**public class** RodCuttingUsingRecursion {

**public static void** main(String[] args) {

**int**[] length = {1,2,3,4};

**int**[] profit = {2,5,9,6};

**int** totalLength = 5;

rodCutRecursion object = **new** rodCutRecursion(length, profit , totalLength);

System.***out***.println(object.findMaxProfit(totalLength));

}

}