```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class BoardGame {
    String name;
    int players;
    int enthusiasm;

    public BoardGame(String name, int players, int enthusiasm) {
        this.name = name;
        this.players = players;
        this.enthusiasm = enthusiasm;
    }
}
public class Main {

    public static int[] findBestSelection(BoardGame[] games, int availableSpace)
{
        int n = games.length;
        int[][] dp = new int[n + 1][availableSpace + 1];

        // Calculando a matriz dp usando programação dinâmica
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= availableSpace; j++) {
                if (games[i - 1].players > j) {
                    dp[i][j] = dp[i - 1][j];
                } else {
                    dp[i][j] = Math.max(dp[i - 1][j], dp[i - 1][j - games[i -
1].players] + games[i - 1].enthusiasm);
                }
            }
        }

        // Reconstruindo a seleção de jogos ótima
        int[] selectedGamesIndices = new int[n];
        int index = 0;
        int i = n;
        int j = availableSpace;
        while (i > 0 && j > 0) {
            if (dp[i][j] != dp[i - 1][j]) {
                selectedGamesIndices[index++] = i - 1;
                j -= games[i - 1].players;
            }
            i--;
        }
        int[] result = new int[index];
        for (int k = 0; k < index; k++) {
            result[k] = selectedGamesIndices[k];
        }
        return result;
    }
    public static int calculateTotalPlayers(BoardGame[] games, int[]
selectedGamesIndices) {
        int totalPlayers = 0;
        for (int index : selectedGamesIndices) {
            totalPlayers += games[index].players;
        }
        return totalPlayers;
    }
    public static int calculateTotalEnthusiasm(BoardGame[] games, int[]
selectedGamesIndices) {
        int totalEnthusiasm = 0;
        for (int index : selectedGamesIndices) {
```

```
            totalEnthusiasm += games[index].enthusiasm;
        }
        return totalEnthusiasm;
    }
    public static void main(String[] args) throws NumberFormatException,
IOException {

        BufferedReader input = new BufferedReader(new
InputStreamReader(System.in));

        int space = Integer.parseInt(input.readLine());
        int boardGames = Integer.parseInt(input.readLine());

        BoardGame[] games = new BoardGame[boardGames];

        for (int i = 0; i < boardGames; i++) {
            String[] gameInfo = input.readLine().split(" ");
            games[i] = new BoardGame(gameInfo[0], Integer.parseInt(gameInfo[1]),
Integer.parseInt(gameInfo[2]));
        }

        int[] selectedGamesIndices = findBestSelection(games, space);

        // Imprimindo o resultado
        System.out.println(selectedGamesIndices.length + " " +
calculateTotalPlayers(games, selectedGamesIndices) + " " +
calculateTotalEnthusiasm(games, selectedGamesIndices));
        for (int i = selectedGamesIndices.length - 1; i >= 0; i--) {
            int index = selectedGamesIndices[i];
            System.out.println(games[index].name);
        }
    }
}
```