let fractionedFilledKnapsack (items : Item array) (knapsack : Knapsack) (takenUntil : int) : int =

let items = items

|> Seq.ofArray

|> Seq.skip takenUntil

|> Seq.toArray

|> Array.sortBy (fun item -> (float32 <| item.ConstraintOf knapsack) / (float32 item.Price))

let length = items.Length

let mutable index = 0

let mutable filled = 0

let mutable totalPrice = 0

while index < length && filled < knapsack.Capacity do

let constr = items.[index].ConstraintOf knapsack

if filled + constr > knapsack.Capacity then

let price = float32 <| items.[index].Price

let rest = float32 <| knapsack.Capacity - filled

let percent = rest / (float32 constr)

totalPrice <- totalPrice + int32 (percent \* price)

index <- length

else

filled <- filled + constr

totalPrice <- totalPrice + items.[index].Price

index <- index + 1

totalPrice