Naloga 3.3

Upam da ni izbrana ta naloga ☺

Peter Zmeda rešuje problem rezanja jeklenih palic v železarni na Jesenicah. Palico celoštevilske dolžine *n* dm je mogoče razrezati na dele celoštevilske dolžine. Nekje je slišal, da bo za problem lahko uporabil pristope dinamičnega programiranja.

a) *Cene različnih dolžin palic na na trgu so:*

dolžina (dm)	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
cena (EUR)	*2*	*4*	*7*	*9*	*10*	*11*	*12*	*19*	*22*

Nekaj podobnega smo delali na vajah

Dolžina	Cena	Dolžina rezov
7	16	3,3,1
8	19	8
9	22	9

b) *Razmere na trgu delovne sile so se spremenile. Pomagajte Petru sestaviti algoritem, če se cene palic na trgu sicer niso spremenile, le:*

1. *za vsak rez palice moramo odšteti 2 EUR; ali pa*
2. *za vsak rez palice moramo odšteti b/a EUR, kjer je b/a razmerje nastalih dolžin ko razrežemo palico dolžine n, ter je a krajša od nastalih dolžin (torej,* $n=a+b$ *in* $a \leq b$ *).*

Za vsakega od obeh primerov podajte cenovno funkcijo ter psevdokodo za izračun optimalnega razreza in iztržene cene. Ciljana časovna zahtevnost algoritmov je $O(n_2)$.

i .)

Cenovna funkcija za palico dolžine i = palice[j].optimalna + palice[i - j].optimalna – 2
Za lažji izračun sem si shranil tudi palico 0. Cene so nastavljene na dejansko ceno palice z dolžino n.
Optimanla bo na začetku in in se potem nakanadno spreminja.

```
For(i=0; i<#palic;i++){
        For(j=i-1; j<i/2;j++)
        If(cena izračunana v tem koraku > max_value)
                Spremenimo ceno palice

Pogledamoše če je cena dražja od optimalne ki smo jo izačunali če je jo zamenjamo;
}
```

ii.)

Cena za palico dolžine I == palica[j].optimala + palica[i-j].optimalna – palica[j].optimalna/palica[i-j].optimalna

Tako napišemo:

```
For(i=0; i<#palic;i++){
        For(j=i-1; j<i/2;j++)
        If(cena izračunana v tem koraku > max_value)
                Spremenimo ceno palice

Pogledamoše če je cena dražja od optimalne ki smo jo izačunali če je jo zamenjamo;
}
```

C.) *Kriza na trgu se je še poglobila in zato ima Peter na voljo računalnik z omejenim pomnilnikom. Pri računu si lahko zapomni zgolj k rešitev podproblemov, kjer je (k < n), ostale podprobleme pa bo moral računati sproti oziroma ponovno. Pri tem predpostavimo, da globina sklada za rekurzivne klice ni omejena.*

Peter bi si sporti shranjeval že izračunane probleme in naenkrat računal samo za dejanski problem, tako bi prihranili ogromno časa za računanje. Vendar bi bilo mogoče pametno omejiti do kam lahko gre rekurzija ker drugače lahko dosežemo mejo pomnilnika in bo vse nehalo delovati