Knapsack:

```
#include <iostream>
 #include <vector>
 #include <unordered_set>
 #include <algorithm>
 using namespace std;
```

a)

```
 int Knapsack_A(vector<int>&S,int k)
{
        int mx = -1;
         unordered_set<int> summ;
         for(auto i:S)
         {
                for (auto j : summ)
                {
                        int p = i + j;
                         if (k >= p)
                         {
                                summ.insert(p);
                                 if (p > k)
                                        mx = p;
                         }
                 }
         }
        return mx;
}
```

b)

```
int Knapsack_B(int k,vector<int>&S)
{
        int mx = 0;
        for (auto i : S)
        {
                if (k >= i + mx)
                         mx = mx + i;
                 else
                        if (i > mx)
                                mx = i;
        }
         return mx;
}
```

Problema este 1/0 Knapsack dar cu conditia ca greutatea unui obiect sa fie egala cu valoarea sa.
Putem rezolva folosind DP,ceea ce rezulta in O(n*weigth) timp si O(2*weigth) spatiu.

Load Balance:

1.
a)
Este suficient sa dam un exemplu.
Pentru valorile {60,60,80},solutia optia este si cea propusa de catre student, afirmatia, astfel, nu este contrazisa.

b)
In acest caz,aproximarea nu este corecta.
Diferenta de incarcatura este 40 in modul.Totusi, daca timpul de lucru este cel mult 10,o solutie optima ar cere o diferenta maxima de 10.
Cum algoritmul este sustinut a fi 1.1 aproximativ,diferenta noastra trebuie sa fie maxim 11.
40>11 => contradictie.

2.
3.

Traveling Salesman Problem:

1.
a)
Reducere la absurd.
Presupunem ca TSP nu este NP-Hard.
Luand doua grafuri G si H,spunem ca muchiile au costul 1 daca exista in G,si costul 2 altfel.
Si graful H indeplineste conditia impusa,iar aplicand TSP,se obtine costul minim=N doar in cazul unui graf hamiltonian G.
Reducem astfel problema la gasirea unui ciclu hamiltonian,un algoritm NP-Hard.
NP!=P implica contradictie.

b)
c)

2.

Vertex Cover: