Problem Statement
Input:

- k : number of expected classes to be openned.
- min : lowerbound of k.
- max : upperbound of k.
- K = 1,..,k: a set of expected classes to be openned.
- N = 1,..,n: a set of classcourse need to be merged.
- w[i]: number of student of of classcourse i, i ∈ N.

Variables:

- x[i] present class which classcourse i will be join in, domain of x[i] is K, i ∈ N.

Invariants:

- sl(j) = $\sum_{i=1}^{n} w[i]$ *where* $x[i] = j$, j ∈ K.
- vi(x[i]): specify how much x[i] violates constraints, vi(x[i]) is non-negative integer.
- s(x): = $\sum_{i=1}^{n} vi(x[i])$, i ∈ N.

Constraints:

- $15 \leq$ sl(j) ≤ 30.

Output:

- Best global solution: x

We solve merging classcourse problem using TabuSearch. We denote:

- x^*: the best global solution.
- vi[x[i] ← v]: is an array in which x[i] is reassigned to v, vi[x[i] ← v] ∈ K.

Input: As problem statement
Output: As problem statement

```
1  k ← min;
2  while k ≤ max do
3  |  InitRandomSolution();
4  |  x* ← x;
5  |  s(x*) ← s(x);
6  |  FindSolutionUsingTabuSearch();
7  |  k ← k + 1;
8  |  if s(x*) = 0 then
9  |  |  return ⟨x*, s(x*)⟩;
10 |  end
11 end
12 return⟨x*, s(x*)⟩;
```

Algorithm 1: FindOptimalSolution();

Input: K = {1,...,k}
Output: A global solution: x

```
1  for i ≤ n do
2  |  x[i] ← randomelementofK;
3  end
```

Algorithm 2: InitRandomSolution()

Input: K = {1,...,k}.
tabu: represents the tabu list.
tbl: length of the tabu list.
nic: number of consecutive iterations that best solution is not improved.
maxStable: if the best solution is not improved after *maxStable* iterations, then the search is restarted.
maxIter: limit of number of iterations.

Output: Best global solution: x

```
1  it ← 0;
2  while it ≤ max do
3  |  F1 ← {x[i] ∈ x | tabu[i] < it ∧ vi(x[i]) is maximal};
4  |  if F1 = ∅ then
5  |  |  InitRandomSolution();
6  |  end
7  |  x[i] ← random element of F1;
8  |  F2 ← {v ∈ K | vi[x[i] ← v] is minimal};
9  |  if F2 = ∅ then
10 |  |  InitRandomSolution();
11 |  else
12 |  |  v ← random element of F2;
13 |  |  x[i] ← v;
14 |  |  if s(x*) < s(x) then
15 |  |  |  s(x*) ← s(x);
16 |  |  |  nic ← 1;
17 |  |  else
18 |  |  |  nic ← nic + 1;
19 |  |  |  if nic > maxStable then
20 |  |  |  |  InitRandomSolution();
21 |  |  |  |  nic ← 1;
22 |  |  |  |  if s(x*) < s(x) then
23 |  |  |  |  |  x* ← x;
24 |  |  |  |  |  s(x*) ← s(x);
25 |  |  |  |  end
26 |  |  |  end
27 |  |  end
28 |  |  tabu[i] ← it + tbl;
29 |  end
30 end
```

Algorithm 3: FindFeasibleSolutionUsingTabuSearch();