```
### Simulated Annealing ###

#This function generates a solution using First Fit Algorithm.
#Once the solution is generated, the energy is caculated.
#In this case, the energy can be described as the number of bins.
get.energy = function(X, weight, V){
  unpacked.items <- vector()
  topack.items = X
  bins = 0
  packed.items = 0
  curr_bin = 0
  while (packed.items < 10) {
    bins = bins + 1
    curr_bin = 0
    for (i in 1:length(topack.items)) {
      if (curr_bin + weight[topack.items[i]] <= V) {
        curr_bin = curr_bin + weight[topack.items[i]]
        curr_bin
        packed.items = packed.items + 1
      } else {
        unpacked.items = cbind(unpacked.items, topack.items[i])
      }
    }
    topack.items = unpacked.items
    unpacked.items <- vector()
  }

  bins
}

#This function generates a new neighbor.
#A neighbor of the current design vector is the current design vector with two items swapped.
get.neighbor = function(X){
  swap = sample(1:10, 2, replace=F)
  replace(X,c(swap[1], swap[2]), X[c(swap[2], swap[1])])
}

#Initializing parameters
T_j = 1000
schedule = 10
cooler = 0.8
max.iter = 10

weight = c(10,20,30,30,40,50,50,50,70,80)
V = 100

X_curr = c(1,2,3,4,5,6,7,8,9,10)
X_best = X_curr
E_curr = get.energy(X_curr, weight, V)
E_best = E_curr

schedule.iter = 0
iter = 0
E_list = NULL
X_list = X_curr
X_best_list = X_curr

#This is the main method that calls all other functions.
while(iter < max.iter){
  #First a neighbor is generated.
  X_i = get.neighbor(X_curr)

  #Here the energy function is called.
  #The energy value is then appended to the E_list.
  #The delta is then caculated.
  E_i = get.energy(X_i, weight, V)
  E_list = c(E_list, E_i)
  delta =  E_i - E_curr

  #If delta is negative, it means the energy of the neighbor is better than the current energy. Therefore, the neighbor becomes the new cu
  if(delta <= 0){
    X_curr = X_i
    E_curr = E_i
  }
  #If delta is positive, it means the energy of the neighbor is worse than the current energy. Therefore, the current design vector remain
  if(delta > 0){
    check = runif(1)
    if(check < exp(-delta/T_j)){
      X_curr = X_i
      E_curr = E_i
    }
  }

  if (E_curr >= E_best) {
    E_best = E_curr
    X_best = X_curr
  }

  schedule.iter = schedule.iter + 1
  if(schedule.iter == schedule){
    schedule.iter = 0
```

```
    T_j = T_j*cooler
  }
  iter = iter + 1
  X_list = cbind(X_list, X_curr)
}

X_best
E_best
```