F# In-Class Example

1. **Open and clear https://try.fsharp.org/**
2. **Copy the provided framework code into the editor**

Code Framework:

```
open System

let names_list = ["Gatchina Palace"; "Clover Leaf"; "Empire Nephrite"; "Peter the Great";
"Moscow Kremlin"; "Alexander Palace"; "Alexander III Commemorative"; "Catherine the
Great"; "Constellation"; "Napoleonic"; "Romanov Tercentenary" ; "Red Cross with Triptych"]

let volumes_list = [2;4;5;2;3;10;5;3;3;9;2;1]
let values_list = [3;19;17;15;13;3;2;8;5;6;17;15]
let maxVolume = 20
let mutable chosen = Set.empty  // for final output; ignore when coding


let rec knapsackProblem (restVol : int)  (i : int) =

//YOUR CODE HERE

let result = knapsackProblem maxVolume 0 //creates an instance of the knapsack

(*The rest of this jumble is to print the solution to this instance of the Knapsack Problem, then
print your code's solution *)

printfn "Correct output: \n \n Name: Clover Leaf Value: 19 Weight: 4 \n Name: Empire Nephrite
Value: 17 Weight: 5 \n Name: Peter the Great Value: 15 Weight: 2 \n Name: Moscow Kremlin
Value: 13 Weight: 3 \n Name: Alexander III Commemorative Value: 2 Weight: 5 \n Name:
Catherine the Great Value: 8 Weight: 3 \n Name: Constellation Value: 5 Weight: 3 \n Name:
Napoleonic Value: 6 Weight: 9 \n Name: Romanov Tercentenary Value: 17 Weight: 2 \n Name:
Red Cross with Triptych Value: 15 Weight: 1"

printfn "\n Your output: \n "

Set.iter (fun i -> printfn "Name:  %s  Value:  %i  Weight:  %i" names_list.[i] values_list.[i]
volumes_list.[i]) chosen
```

3. **Provide code that solves the knapsack problem**

Important stuff
rec : F# explicitly calls recursion using rec
Example: let rec funcName ...

Complete the Knapsack problem using (or not using) the code provided and the following specification:

- Knapsack has a maximum volume, it cannot contain any item that would increase the cumulative volume of the bag over this limit (our example sets a limit of 20)
- Each ‘item’ being considered has its own volume, and a corresponding value. The objective is to choose the ideal subset of items that both maximize value and stay under the maximum volume.

Key:

```
open System

let names_list = ["Gatchina Palace"; "Clover Leaf"; "Empire Nephrite"; "Peter the Great";
"Moscow Kremlin"; "Alexander Palace"; "Alexander III Commemorative"; "Catherine the
Great"; "Constellation"; "Napoleonic"; "Romanov Tercentenary" ; "Red Cross with Triptych"]

let volumes_list = [2;4;5;2;3;10;5;3;3;9;2;1]
let values_list = [3;19;17;15;13;3;2;8;5;6;17;15]
let maxVolume = 20
let mutable chosen = Set.empty  // for final output; ignore when coding

let rec knapsackProblem (restVol : int)  (i : int) =

(* YOUR CODE *)
   if i < volumes_list.Length then
      let toss = knapsackProblem(restVol )(i+1)
      let keep =
         if restVol - volumes_list.[i] >= 0 then
            values_list.[i] + knapsackProblem(restVol - volumes_list.[i]) (i+1)
         else 0

      if toss > keep then
            toss
      else
            chosen <- chosen.Add(i)
            keep
   else
      0



let result = knapsackProblem maxVolume 0

Set.iter (fun i -> printfn "Name:  %s  Value:  %i  Weight:  %i" names_list.[i] values_list.[i]
volumes_list.[i]) chosen
```

Sources:
Knapsack Problem in F# with recursive function
https://stackoverflow.com/questions/62432616/knapsack-problem-in-f-with-recursive-function