1. O algoritmo abaixo apresenta uma solução recursiva para o problema Knapsack.

```
knapsack(int capacity, int i) {
        if (capacity <= 0 OU i < 0) then
                return 0
        int value_if_i_not_picked = knapsack(capacity, i - 1)
        int value_if_i_picked = 0
        if (capacity >= Weights[i]) then
                //Values é um array que guarda o valor de cada item
                //Weights é um array que guarda o peso de cada item
                value_if_i_picked = Values[i] + knapsack(capacity - Weights[i], i – 1)
        //função max retorna o maior entre os dois elementos
        return max(value_if_i_picked, value_if_i_not_picked)
}
```

   a) Faça as alterações necessárias para que o algoritmo utilize a abordagem top-down (com memoização) de programação dinâmica.

   b) Faça as alterações necessárias para que o algoritmo utilize a abordagem bottom-up de programação dinâmica. Use memoização nesse caso.

2. Uma solução recursiva para o problema da *Subsequência Crescente Mais Longa* (*SCML*) é dada abaixo.

```
maiorSCML(int n, int num[], int i) {
        if (i = n) then
                return 1
        m ← 1
        for j ← i+1 to n do
                if (num[j] > num[i]) then
                        m ← max(m, 1 + maiorSCML(n, num, j))
        return m
}
```

   a) Não está sendo utilizada nenhuma memoização, identifique como usar a memoização e implemente uma nova versão top-down desse algoritmo mas com memoização.

   b) Faça também a implementação iterativa bottom-up deste algoritmo que também se aproveita da memoização.