a) Give a written description and pseudo-code for your algorithm. Try to create an algorithm that is efficient in both time and storage requirements.

1: each family member has run their own knapsack function.

2: after each family member ran the function, delete these items from the shop list.

3: repeat until all family members' chars are full.

```
for familyMemeber in familyMemebers:
        valueList.remove(pickedList's indexs);
        weightLisit.remove(pickedList's indexs);
        pickedIList = Knapsack(totalVolume, valueList, weightLisit, cur_index)


  Knapsack(__max_volume, __wt, __val, cur_index):
    K = [[0 for x in range(__max_volume + 1)] for x in range(cur_index + 1)]
    for i in range(cur_index + 1):
      for w in range(__max_volume+1):
        if i == 0 or w == 0:
          K[i][w] = 0
        elif __wt[i-1] <= w:
          K[i][w] = max(__val[i-1] + K[i-1][w-__wt[i-1]],  K[i-1][w])
        else:
          K[i][w] = K[i-1][w]
    return K[cur_index][__max_volume]
```