Team Members:

1) Mohamed Diaa Abdelaal 43-12821 t7

2) Abdelrahman Gelany 43-17100 t9

3) Mohamed Ahmed Abdelmaksoud 43-16710 t9

English description to our code flow.

1) The Knapsack Method.

```java
public static void KnapSackMethod(int W, ArrayList<Integer> wk, ArrayList<Integer> bk, int length) {
    int towD[][] = new int[length + 1][W + 1];

    for (int i = 0; i <= length; i++) {
        for (int j = 0; j <= W; j++) {
            if (i == 0 || j == 0) {
                towD[i][j] = 0;
            } else if (wk.get(i - 1) > j) {
                towD[i][j] = towD[i - 1][j];
            }

            else {
                towD[i][j] = max(towD[i - 1][j],bk.get(i - 1) + towD[i - 1][j - wk.get(i - 1)]);
            }
        }
    }

    int res = towD[length][W];
    System.out.println("\n" + "The result = " + res + "\n");

    int w = W;
    for (int i = length; i > 0 && res > 0; i--) {

        if (res == towD[i - 1][w])
            continue;
        else {

            int index = wk.indexOf(wk.get(i - 1)) + 1;
            System.out.print("item " + "i" + index + ", W = " + wk.get(i - 1) +"."+ "\n");

            res = res - bk.get(i - 1);
            w = w - wk.get(i - 1);
        }
    }
```

This nested loop does the same table as in the tutorial

This loop gets the last cell (the maximum value) and also finds the selected items

The description for this method: (I) we enter the method with four values Width, arraylist of the weights, arraylist of the benefits and the length of the list. Then we make a 2D array to be like the table in the tutorial which has all the values in it. We made a nested loop to loop over the two arrays to fill the 2D array with the wanted values by making the following: if the index is zero or the weight is zero then fill this cell by zero, else if the wk>W then this is the first case in the tutorial so we fill the cell with towD[i-1][j], else this is the second case in the tutorial so we fill the cell with the maximum of the upper cell (towD[i-1]) and the summation of the benefit and the other cell (bk.get(i-1)+towD[i-1][j-wk.get(i-1)]).

$$B(k,w) = \begin{cases} B(k-1,w) & \text{if } w_k > w \\ max(B(k-1,w), B(k-1,w-w_k) + b_k) & \text{otherwise} \end{cases}$$

(II) After the nested loop we print the largest value which is the last cell, after that we made another loop to get the items which makes the following: check if the current cell value is equal to the upper cell then this item will not be included the else part means that we will take this cell's value and print it.

```
56
57      public static void main(String[] args) throws IOException {
58          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
59          System.out.println("Enter the pack Weight (W) !!");
60          int W = Integer.parseInt(br.readLine());
61          System.out.println("\n" + "Enter the Weight of the items (wk) !!");
62          StringTokenizer st = new StringTokenizer(br.readLine());
63          ArrayList<Integer> wk = new ArrayList<>();
64          while (st.hasMoreTokens()) {
65              int x = Integer.parseInt(st.nextToken());
66              wk.add(x);
67          }
68          System.out.println("\n" + "Enter the benefit of the items (bk) !!");
69          StringTokenizer st2 = new StringTokenizer(br.readLine());
70          ArrayList<Integer> bk = new ArrayList<>();
71          while (st2.hasMoreTokens()) {
72              int x = Integer.parseInt(st2.nextToken());
73              bk.add(x);
74          }
75          int length = bk.size();
76          KnapSackMethod(W, wk, bk, length);
77
78      }
79  }
80
```

Here we use bufferedReader&StringTokenizer to fill in the arrays and finally call the method to print the wanted results

2) The main Method.

First we make the buffered reader to take the data from the user, and then we take the input line by line using string tokenizer and then we made loops to fill the arrays

After that we call the method and pass the parameters and then it will print the maximum value and the items.

3) The output.

KnapSackMasterAlgorithm.java

```
                towD[i][j] = max(towD[i - 1][j],bk.get(i - 1) + towD[i - 1][j - wk.get(i - 1)]);
            }
        }
    }

    int res = towD[length][W];
    System.out.println("\n" + "The result = " + res + "\n");

    int w = W;
    for (int i = length; i > 0 && res > 0; i--) {

        if (res == towD[i - 1][w])
            continue;
        else {

            int index = wk.indexOf(wk.get(i - 1)) + 1;
            System.out.print("item " + "i" + index + ", W = " + wk.get(i - 1) +"."+ "\n");
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Console

<terminated> KnapSackMasterAlgorithm [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Dec 30, 2020, 12:57:06 PM)

```
Enter the pack Weight (W) !!
10

Enter the Weight of the items (wk) !!
3 4 6 1 7

Enter the benefit of the items (bk) !!
4 7 8 2 10

The result = 15

item i3, W = 6.
item i2, W = 4.
```