Math 381 A
Sheng Yu

# Homework #1 Writeup: Knapsack

## I. Introduction:

A group of thieves named by the Suicide Squad decide to rob a local jewelry store where inside the jewelry store there are total 50 different types of jewelry. The shapes, volumes and weights of the jewelries are varied, so they have prepared a vehicle with a weight capacity of 200 kg and a volume capacity of 100 liters to help them move the jewelries. The Suicide Squad leader named by Mr. X has took Math 381 at UW before, so he knows Linear Programming well. Before the heist, Mr. X has gathered some information with the value, weight, and volume information of the jewelries in that store, and now he is planning to make the most profit from this heist by using Linear Programming.

## II. Linear Programming:

**Part (a):**

With the information gathered, here are the information with the 50 different types of jewelry:

$$Jewelry\ i, where\ 1 \leq i \leq 50\ has:$$

$$\begin{cases} value = floor(50 + 30\cos(i)) & thousands\ of\ dollars \\ weight = floor(14 + 9\cos(11i + 2)) & kg \\ volume = floor(10 + 2\cos(4i - 1)) & liters \end{cases}$$

Suppose each jewelry is unique and a Boolean value $x_i$ is used where $x_i \in \{0, 1\}$, with 1 meaning Mr. X decides to take that piece of jewelry, and 0 otherwise. Based on the vehicle volume and weight capacity, Mr. X gets the following LP:

$Maximize:$

$$\sum_{i=1}^{50} x_i \cdot floor(50 + 30\cos(i))$$

$Subject\ to:$

$$\sum_{i=1}^{50} x_i \cdot\ floor(14 + 9\cos(11i + 2))\ \leq 200$$

$$\sum_{i=1}^{50} x_i \cdot floor(10 + 2\cos(4i - 1))\ \leq 100$$

$$x_i \in \{0, 1\}, where\ 1 \leq i \leq 50$$

Mr. X wrote the following code into MATLAB to generate the LPSolve input file:

```
% Set up

value_list = floor(50 + 30 * cos(1:50));
weight_list = floor(14 + 9 * cos(11 * (1:50) + 2));
volume_list = floor(10 + 2 * cos(4 * (1:50) - 1));

% Generate the lpsolve input file

lp = mxlpsolve('make_lp', 0, 50);
mxlpsolve('set_verbose', lp, 3);

mxlpsolve('set_maxim', lp);                             % Maximize
mxlpsolve('set_binary', lp, (1:50));                    % Set xi to be 0 or 1
mxlpsolve('set_obj_fn', lp, value_list);                % Set object fun

mxlpsolve('add_constraint', lp, weight_list, 1, 200);   % Total weight <= 200 kg
mxlpsolve('add_constraint', lp, volume_list, 1, 100);   % Total volume <= 100 L

mxlpsolve('set_row_name', lp, 1, 'Weight');
mxlpsolve('set_row_name', lp, 2, 'Volume');

mxlpsolve('write_lp', lp, 'hw1a.lp');
```

And this produced the LPSolve input file: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
/* Objective function */
max: +66 C1 +37 C2 +20 C3 +30 C4 +58 C5 +78 C6 +72 C7 +45 C8 +22 C9 +24 C10 +50 C11 +75 C12 +77 C13 +54 C14
 +27 C15 +21 C16 +41 C17 +69 C18 +79 C19 +62 C20 +33 C21 +20 C22 +34 C23 +62 C24 +79 C25 +69 C26 +41 C27
 +21 C28 +27 C29 +54 C30 +77 C31 +75 C32 +49 C33 +24 C34 +22 C35 +46 C36 +72 C37 +78 C38 +57 C39 +29 C40
 +20 C41 +38 C42 +66 C43 +79 C44 +65 C45 +37 C46 +20 C47 +30 C48 +59 C49 +78 C50;

/* Constraints */
Weight: +22 C1 +17 C2 +5 C3 +10 C4 +22 C5 +17 C6 +5 C7 +9 C8 +22 C9 +18 C10 +6 C11 +9 C12 +21 C13 +18 C14 +6 C15
 +9 C16 +21 C17 +18 C18 +6 C19 +9 C20 +21 C21 +18 C22 +6 C23 +9 C24 +21 C25 +18 C26 +6 C27 +9 C28 +21 C29
 +18 C30 +6 C31 +9 C32 +21 C33 +18 C34 +6 C35 +9 C36 +21 C37 +19 C38 +6 C39 +8 C40 +21 C41 +19 C42 +6 C43
 +8 C44 +21 C45 +19 C46 +6 C47 +8 C48 +21 C49 +19 C50 <= 200;
Volume: +8 C1 +11 C2 +10 C3 +8 C4 +11 C5 +8 C6 +9 C7 +11 C8 +8 C9 +10 C10 +11 C11 +8 C12 +11 C13 +10 C14 +8 C15
 +11 C16 +8 C17 +9 C18 +11 C19 +8 C20 +10 C21 +11 C22 +8 C23 +11 C24 +10 C25 +8 C26 +11 C27 +8 C28 +9 C29
 +11 C30 +8 C31 +10 C32 +11 C33 +8 C34 +11 C35 +10 C36 +8 C37 +11 C38 +9 C39 +9 C40 +11 C41 +8 C42 +10 C43
 +11 C44 +8 C45 +11 C46 +10 C47 +8 C48 +11 C49 +9 C50 <= 100;

/* Variable bounds */
C1 <= 1;C2 <= 1;C3 <= 1;C4 <= 1;C5 <= 1;C6 <= 1;C7 <= 1;C8 <= 1;C9 <= 1;C10 <= 1;C11 <= 1;C12 <= 1;C13 <= 1;C14 <= 1;
C15 <= 1;C16 <= 1;C17 <= 1;C18 <= 1;C19 <= 1;C20 <= 1;C21 <= 1;C22 <= 1;C23 <= 1;C24 <= 1;C25 <= 1;C26 <= 1;C27 <= 1;
C28 <= 1;C29 <= 1;C30 <= 1;C31 <= 1;C32 <= 1;C33 <= 1;C34 <= 1;C35 <= 1;C36 <= 1;C37 <= 1;C38 <= 1;C39 <= 1;C40 <= 1;
C41 <= 1;C42 <= 1;C43 <= 1;C44 <= 1;C45 <= 1;C46 <= 1;C47 <= 1;C48 <= 1;C49 <= 1;C50 <= 1;

/* Integer definitions */
int C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32,
    C33,C34,C35,C36,C37,C38,C39,C40,C41,C42,C43,C44,C45,C46,C47,C48,C49,C50;
```

Then Mr. X gets the following output: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
Value of objective function: 840.00000000


Actual values of the variables:

C1                        1

C6                        1

C7                        1

C12                       1

C18                       1

C20                       1

C26                       1

C31                       1

C37                       1

C39                       1

C45                       1

C50                       1
```

From the output, Mr. X knows to get the most value, he will need to take jewelries with number of $i$, where:

$$i = 1, 6, 7, 12, 18, 20, 26, 31, 37, 39, 45, 50 \qquad \text{(12 Pieces of Jewelry)}$$

$$Total\ Value = 840\ thousands\ of\ dollars$$

**Part (b) - 1:**

However, after getting the result, Mr. X gets new information that the jewelries with prime indexes are more popular on the black market which can be sold in a much shorter time, so he decides the prime-indexed jewelries, where:

$$Prime\ Indexes\ (PI) = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47\}$$

must make up at least 50% of the total values of all the jewelries he takes.

Let $Not\ Prime\ Indexes\ (NPI)$ denotes the rest indexes from 1 to 50 that are not prime numbers, then the updated LP would be:

$Maximize$:

$$\sum_{i=1}^{50} x_i \cdot floor(50 + 30\cos(i))$$

$Subject\ to$:

$$\sum_{i=1}^{50} x_i \cdot\ floor(14 + 9\cos(11i + 2))\ \leq 200$$

$$\sum_{i=1}^{50} x_i \cdot floor(10 + 2\cos(4i - 1))\ \leq 100$$

$$\sum_{i\ in\ PI} x_i \cdot floor(50 + 30\cos(i)) \geq \frac{1}{2} * \sum_{i=1}^{50} x_i \cdot floor(50 + 30\cos(i)) \qquad (New\ Constraint)$$

$$x_i \in \{0, 1\}, where\ 1 \leq i \leq 50$$

We can rewrite the $New\ Constraint$, cancel the same elements on both side:

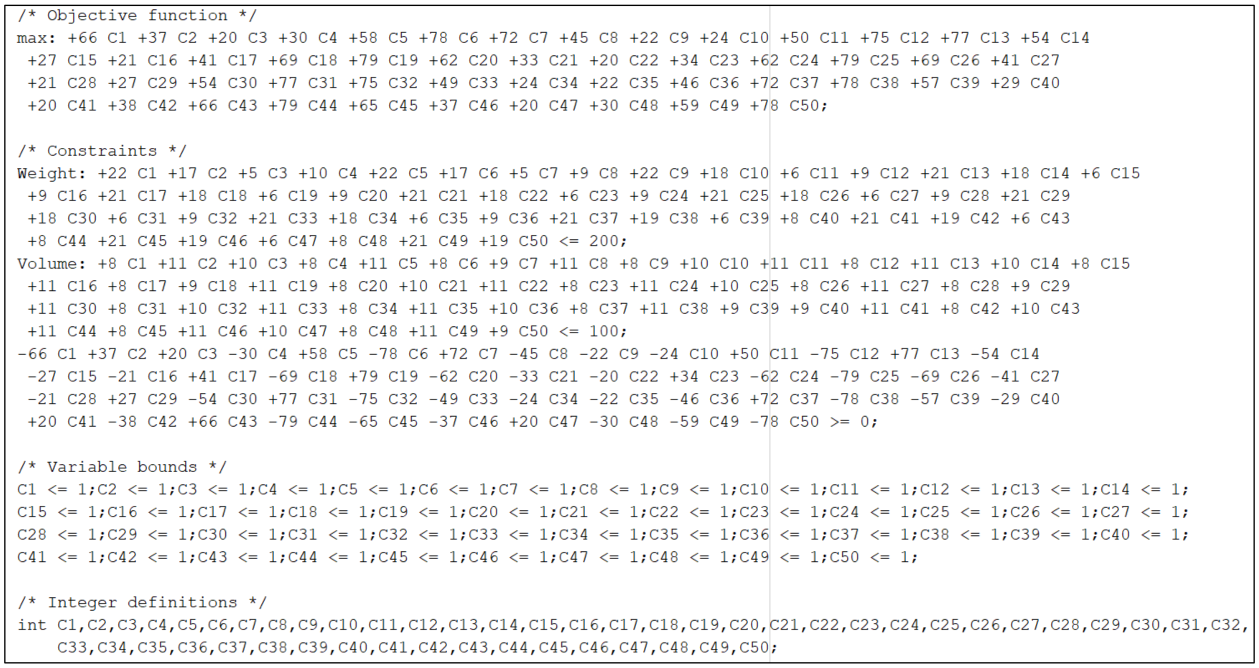
$$\frac{1}{2}\cdot\sum_{i\ in\ PI} x_i\cdot floor(50+30\cos(i)) - \frac{1}{2}\cdot\sum_{i\ in\ NPI} x_i\cdot floor(50+30\cos(i)) \geq 0$$

$$\Rightarrow \sum_{i\ in\ PI} x_i\cdot floor(50+30\cos(i)) - \sum_{i\ in\ NPI} x_i\cdot floor(50+30\cos(i)) \geq 0$$

Here is MATLAB code Mr. X wrote to add this constraint:

```
% Set up

value_list = floor(50 + 30 * cos(1:50));
weight_list = floor(14 + 9 * cos(11 * (1:50) + 2));
volume_list = floor(10 + 2 * cos(4 * (1:50) - 1));

% Make the list for the third newly added constraint

PI = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47];
PI_value = value_list;

for i = 1:50
    if ismember(i, PI) == 0
        PI_value(i) = 0;
    end
end

NPI_value = value_list - PI_value;
new_cons_list = PI_value - NPI_value;


% Generate the lpsolve input file

lp = mxlpsolve('make_lp', 0, 50);
mxlpsolve('set_verbose', lp, 3);

mxlpsolve('set_maxim', lp);                               % Maximize
mxlpsolve('set_binary', lp, (1:50));                      % Set xi to be 0 or 1
mxlpsolve('set_obj_fn', lp, value_list);                  % Set object fun

mxlpsolve('add_constraint', lp, weight_list, 1, 200);     % Total weight <= 200 kg
mxlpsolve('add_constraint', lp, volume_list, 1, 100);     % Total volume <= 100 L
mxlpsolve('add_constraint', lp, new_cons_list, 2, 0);     % Prime Indexed value at least 50%

mxlpsolve('set_row_name', lp, 1, 'Weight');
mxlpsolve('set_row_name', lp, 2, 'Volume');

mxlpsolve('write_lp', lp, 'hw1b1.lp');
```

And this produced the LPSolve input file: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)



```
/* Objective function */
max: +66 C1 +37 C2 +20 C3 +30 C4 +58 C5 +78 C6 +72 C7 +45 C8 +22 C9 +24 C10 +50 C11 +75 C12 +77 C13 +54 C14
 +27 C15 +21 C16 +41 C17 +69 C18 +79 C19 +62 C20 +33 C21 +20 C22 +34 C23 +62 C24 +79 C25 +69 C26 +41 C27
 +21 C28 +27 C29 +54 C30 +77 C31 +75 C32 +49 C33 +24 C34 +22 C35 +46 C36 +72 C37 +78 C38 +57 C39 +29 C40
 +20 C41 +38 C42 +66 C43 +79 C44 +65 C45 +37 C46 +20 C47 +30 C48 +59 C49 +78 C50;

/* Constraints */
Weight: +22 C1 +17 C2 +5 C3 +10 C4 +22 C5 +17 C6 +5 C7 +9 C8 +22 C9 +18 C10 +6 C11 +9 C12 +21 C13 +18 C14 +6 C15
 +9 C16 +21 C17 +18 C18 +6 C19 +9 C20 +21 C21 +18 C22 +6 C23 +9 C24 +21 C25 +18 C26 +6 C27 +9 C28 +21 C29
 +18 C30 +6 C31 +9 C32 +21 C33 +18 C34 +6 C35 +9 C36 +21 C37 +19 C38 +6 C39 +8 C40 +21 C41 +19 C42 +6 C43
 +8 C44 +21 C45 +19 C46 +6 C47 +8 C48 +21 C49 +19 C50 <= 200;
Volume: +8 C1 +11 C2 +10 C3 +8 C4 +11 C5 +8 C6 +9 C7 +11 C8 +8 C9 +10 C10 +11 C11 +8 C12 +11 C13 +10 C14 +8 C15
 +11 C16 +8 C17 +9 C18 +11 C19 +8 C20 +10 C21 +11 C22 +8 C23 +11 C24 +10 C25 +8 C26 +11 C27 +8 C28 +9 C29
 +11 C30 +8 C31 +10 C32 +11 C33 +8 C34 +11 C35 +10 C36 +8 C37 +11 C38 +9 C39 +9 C40 +11 C41 +8 C42 +10 C43
 +11 C44 +8 C45 +11 C46 +10 C47 +8 C48 +11 C49 +9 C50 <= 100;
-66 C1 +37 C2 +20 C3 -30 C4 +58 C5 -78 C6 +72 C7 -45 C8 -22 C9 -24 C10 +50 C11 -75 C12 +77 C13 -54 C14
 -27 C15 -21 C16 +41 C17 -69 C18 +79 C19 -62 C20 -33 C21 -20 C22 +34 C23 -62 C24 -79 C25 -69 C26 -41 C27
 -21 C28 +27 C29 -54 C30 +77 C31 -75 C32 -49 C33 -24 C34 -22 C35 -46 C36 +72 C37 -78 C38 -57 C39 -29 C40
 +20 C41 -38 C42 +66 C43 -79 C44 -65 C45 -37 C46 +20 C47 -30 C48 -59 C49 -78 C50 >= 0;

/* Variable bounds */
C1 <= 1;C2 <= 1;C3 <= 1;C4 <= 1;C5 <= 1;C6 <= 1;C7 <= 1;C8 <= 1;C9 <= 1;C10 <= 1;C11 <= 1;C12 <= 1;C13 <= 1;C14 <= 1;
C15 <= 1;C16 <= 1;C17 <= 1;C18 <= 1;C19 <= 1;C20 <= 1;C21 <= 1;C22 <= 1;C23 <= 1;C24 <= 1;C25 <= 1;C26 <= 1;C27 <= 1;
C28 <= 1;C29 <= 1;C30 <= 1;C31 <= 1;C32 <= 1;C33 <= 1;C34 <= 1;C35 <= 1;C36 <= 1;C37 <= 1;C38 <= 1;C39 <= 1;C40 <= 1;
C41 <= 1;C42 <= 1;C43 <= 1;C44 <= 1;C45 <= 1;C46 <= 1;C47 <= 1;C48 <= 1;C49 <= 1;C50 <= 1;

/* Integer definitions */
int C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32,
    C33,C34,C35,C36,C37,C38,C39,C40,C41,C42,C43,C44,C45,C46,C47,C48,C49,C50;
```

Then Mr. X gets the following output: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
Value of objective function: 822.00000000


Actual values of the variables:

C6              1

C7              1

C12             1

C13             1

C19             1

C25             1

C26             1

C31             1

C37             1

C43             1

C50             1
```

From the output, Mr. X knows to get the most value with at least 50% total values comes from prime indexed jewelries, he will need to take jewelry with index of $i$, where:

$$i = 6, 7, 12, 13, 19, 25, 26, 31, 37, 43, 50 \qquad \text{(10 Pieces of Jewelry)}$$

$$Total\ Value = 822\ thousands\ of\ dollars$$

$$Total\ Prime\ Index\ Jewels = 443\ thousands\ of\ dollars$$

$$\frac{443}{822} \times 100\% \approx 53.9\% \ \geq 50\%$$

Mr. X finds out that after adding this new constraint, the total value decreased from 840 thousand of dollars to 822 thousand of dollars, and the total number of pieces decreased from 12 pieces to 10 pieces. Since there are too much risk for such heist, Mr. X prefers the actual profit more than the selling efficiency. But he still wants to see if he can have both.

**Part (b) - 2:**

Mr. X finally made his mind, he wants to see if there is any possibility to have both the profit and the selling efficiency, so he decides the prime-indexed jewelries, where:

$$Prime\ Indexes\ (PI) = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47\}$$

must make up at least 75% of the total values of all the jewelries he takes.

The updated LP would be:

$Maximize$:

$$\sum_{i=1}^{50} x_i \cdot floor(50 + 30\cos(i))$$

$Subject\ to$:

$$\sum_{i=1}^{50} x_i \cdot\ floor(14 + 9\cos(11i + 2))\ \leq 200$$

$$\sum_{i=1}^{50} x_i \cdot floor(10 + 2\cos(4i - 1))\ \leq 100$$

$$\sum_{i\ in\ PI} x_i \cdot floor(50 + 30\cos(i)) \geq \frac{3}{4} \cdot \sum_{i=1}^{50} x_i \cdot floor(50 + 30\cos(i)) \qquad (New\ Constraint)$$

$$x_i \in \{0, 1\}, where\ 1 \leq i \leq 50$$

We can rewrite the $New\ Constraint$, cancel the same elements on both side:

$$\frac{1}{4} \cdot \sum_{i\ in\ PI} x_i \cdot floor(50 + 30\cos(i)) - \frac{3}{4} \cdot \sum_{i\ in\ NPI} x_i \cdot floor(50 + 30\cos(i)) \geq 0$$

$$\Rightarrow \quad \sum_{i\ in\ PI} x_i \cdot floor(50 + 30\cos(i)) - 3 \cdot \sum_{i\ in\ NPI} x_i \cdot floor(50 + 30\cos(i)) \geq 0$$

Here is MATLAB code Mr. X wrote to add this constraint:

```matlab
% Set up

value_list = floor(50 + 30 * cos(1:50));
weight_list = floor(14 + 9 * cos(11 * (1:50) + 2));
volume_list = floor(10 + 2 * cos(4 * (1:50) - 1));

% Make the list for the third newly added constraint

PI = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47];
PI_value = value_list;

for i = 1:50
    if ismember(i, PI) == 0
        PI_value(i) = 0;
    end
end

NPI_value = value_list - PI_value;
new_cons_list = PI_value - 3 * NPI_value;


% Generate the lpsolve input file

lp = mxlpsolve('make_lp', 0, 50);
mxlpsolve('set_verbose', lp, 3);

mxlpsolve('set_maxim', lp);                             % Maximize
mxlpsolve('set_binary', lp, (1:50));                    % Set xi to be 0 or 1
mxlpsolve('set_obj_fn', lp, value_list);                % Set object fun

mxlpsolve('add_constraint', lp, weight_list, 1, 200);   % Total weight <= 200 kg
mxlpsolve('add_constraint', lp, volume_list, 1, 100);   % Total volume <= 100 L
mxlpsolve('add_constraint', lp, new_cons_list, 2, 0);   % Prime Indexed value at least 50%

mxlpsolve('set_row_name', lp, 1, 'Weight');
mxlpsolve('set_row_name', lp, 2, 'Volume');

mxlpsolve('write_lp', lp, 'hw1b2.lp');
```

And this produced the LPSolve input file: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
/* Objective function */
max: +66 C1 +37 C2 +20 C3 +30 C4 +58 C5 +78 C6 +72 C7 +45 C8 +22 C9 +24 C10 +50 C11 +75 C12 +77 C13 +54 C14
 +27 C15 +21 C16 +41 C17 +69 C18 +79 C19 +62 C20 +33 C21 +20 C22 +34 C23 +62 C24 +79 C25 +69 C26 +41 C27
 +21 C28 +27 C29 +54 C30 +77 C31 +75 C32 +49 C33 +24 C34 +22 C35 +46 C36 +72 C37 +78 C38 +57 C39 +29 C40
 +20 C41 +38 C42 +66 C43 +79 C44 +65 C45 +37 C46 +20 C47 +30 C48 +59 C49 +78 C50;

/* Constraints */
Weight: +22 C1 +17 C2 +5 C3 +10 C4 +22 C5 +17 C6 +5 C7 +9 C8 +22 C9 +18 C10 +6 C11 +9 C12 +21 C13 +18 C14 +6 C15
 +9 C16 +21 C17 +18 C18 +6 C19 +9 C20 +21 C21 +18 C22 +6 C23 +9 C24 +21 C25 +18 C26 +6 C27 +9 C28 +21 C29
 +18 C30 +6 C31 +9 C32 +21 C33 +18 C34 +6 C35 +9 C36 +21 C37 +19 C38 +6 C39 +8 C40 +21 C41 +19 C42 +6 C43
 +8 C44 +21 C45 +19 C46 +6 C47 +8 C48 +21 C49 +19 C50 <= 200;
Volume: +8 C1 +11 C2 +10 C3 +8 C4 +11 C5 +8 C6 +9 C7 +11 C8 +8 C9 +10 C10 +11 C11 +8 C12 +11 C13 +10 C14 +8 C15
 +11 C16 +8 C17 +9 C18 +11 C19 +8 C20 +10 C21 +11 C22 +8 C23 +11 C24 +10 C25 +8 C26 +11 C27 +8 C28 +9 C29
 +11 C30 +8 C31 +10 C32 +11 C33 +8 C34 +11 C35 +10 C36 +8 C37 +11 C38 +9 C39 +9 C40 +11 C41 +8 C42 +10 C43
 +11 C44 +8 C45 +11 C46 +10 C47 +8 C48 +11 C49 +9 C50 <= 100;
-198 C1 +37 C2 +20 C3 -90 C4 +58 C5 -234 C6 +72 C7 -135 C8 -66 C9 -72 C10 +50 C11 -225 C12 +77 C13 -162 C14
 -81 C15 -63 C16 +41 C17 -207 C18 +79 C19 -186 C20 -99 C21 -60 C22 +34 C23 -186 C24 -237 C25 -207 C26
 -123 C27 -63 C28 +27 C29 -162 C30 +77 C31 -225 C32 -147 C33 -72 C34 -66 C35 -138 C36 +72 C37 -234 C38
 -171 C39 -87 C40 +20 C41 -114 C42 +66 C43 -237 C44 -195 C45 -111 C46 +20 C47 -90 C48 -177 C49 -234 C50 >= 0;

/* Variable bounds */
C1 <= 1;C2 <= 1;C3 <= 1;C4 <= 1;C5 <= 1;C6 <= 1;C7 <= 1;C8 <= 1;C9 <= 1;C10 <= 1;C11 <= 1;C12 <= 1;C13 <= 1;C14 <= 1;
C15 <= 1;C16 <= 1;C17 <= 1;C18 <= 1;C19 <= 1;C20 <= 1;C21 <= 1;C22 <= 1;C23 <= 1;C24 <= 1;C25 <= 1;C26 <= 1;C27 <= 1;
C28 <= 1;C29 <= 1;C30 <= 1;C31 <= 1;C32 <= 1;C33 <= 1;C34 <= 1;C35 <= 1;C36 <= 1;C37 <= 1;C38 <= 1;C39 <= 1;C40 <= 1;
C41 <= 1;C42 <= 1;C43 <= 1;C44 <= 1;C45 <= 1;C46 <= 1;C47 <= 1;C48 <= 1;C49 <= 1;C50 <= 1;

/* Integer definitions */
int C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32,
    C33,C34,C35,C36,C37,C38,C39,C40,C41,C42,C43,C44,C45,C46,C47,C48,C49,C50;
```

Then Mr. X gets the following output: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
Value of objective function: 729.00000000


Actual values of the variables:
C5                  1
C6                  1
C7                  1
C12                 1
C13                 1
C17                 1
C19                 1
C23                 1
C31                 1
C37                 1
C43                 1
```

From the output, Mr. X knows to get the most value with at least 75% total values comes from prime indexed jewelries, he will need to take jewelry with index of $i$, where:

$$i = 5, 6, 7, 12, 13, 17, 19, 23, 31, 37, 43 \qquad \text{(11 Pieces of Jewelry)}$$

$$Total\ Value = 729\ thousands\ of\ dollars$$

$$Total\ Prime\ Index\ Jewels = 576\ thousands\ of\ dollars$$

$$\frac{576}{729} \times 100\% \approx 79.0\% \ \geq 75\%$$

Mr. X finds out that after adding this new constraint, the total value decreased from 840 thousand of dollars to 729 thousand of dollars, but the total number of pieces decreased from 12 pieces to 11 pieces. While Mr. X is still trying to judge between the profit and the selling efficiency, he gets some new information again that the jewelries are not unique. There can be multiple identical jewelries.

**Part (c):**

With the given new information, Mr. X has modified the LP as following:

*Maximize*:

$$\sum_{i=1}^{50} x_i \cdot floor(50 + 30\cos(i))$$

*Subject to*:

$$\sum_{i=1}^{50} x_i \cdot \; floor(14 + 9\cos(11i + 2)) \; \leq 200$$

$$\sum_{i=1}^{50} x_i \cdot floor(10 + 2\cos(4i - 1)) \; \leq 100$$

$$x_i \geq 0 \textit{ and } x_i \textit{ is an integer, where } 1 \leq i \leq 50$$

Here is MATLAB code Mr. X wrote:

```
% Set up

value_list = floor(50 + 30 * cos(1:50));
weight_list = floor(14 + 9 * cos(11 * (1:50) + 2));
volume_list = floor(10 + 2 * cos(4 * (1:50) - 1));

% Generate the lpsolve input file

lp = mxlpsolve('make_lp', 0, 50);
mxlpsolve('set_verbose', lp, 3);

mxlpsolve('set_maxim', lp);                             % Maximize
mxlpsolve('set_int', lp, (1:50));                       % Set xi >= 0 and is integer
mxlpsolve('set_obj_fn', lp, value_list);                % Set object fun

mxlpsolve('add_constraint', lp, weight_list, 1, 200);   % Total weight <= 200 kg
mxlpsolve('add_constraint', lp, volume_list, 1, 100);   % Total volume <= 100 L

mxlpsolve('set_row_name', lp, 1, 'Weight');
mxlpsolve('set_row_name', lp, 2, 'Volume');

mxlpsolve('write_lp', lp, 'hw1c.lp');
```

And this produced the LPSolve input file: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
/* Objective function */
max: +66 C1 +37 C2 +20 C3 +30 C4 +58 C5 +78 C6 +72 C7 +45 C8 +22 C9 +24 C10 +50 C11 +75 C12 +77 C13 +54 C14
 +27 C15 +21 C16 +41 C17 +69 C18 +79 C19 +62 C20 +33 C21 +20 C22 +34 C23 +62 C24 +79 C25 +69 C26 +41 C27
 +21 C28 +27 C29 +54 C30 +77 C31 +75 C32 +49 C33 +24 C34 +22 C35 +46 C36 +72 C37 +78 C38 +57 C39 +29 C40
 +20 C41 +38 C42 +66 C43 +79 C44 +65 C45 +37 C46 +20 C47 +30 C48 +59 C49 +78 C50;

/* Constraints */
Weight: +22 C1 +17 C2 +5 C3 +10 C4 +22 C5 +17 C6 +5 C7 +9 C8 +22 C9 +18 C10 +6 C11 +9 C12 +21 C13 +18 C14 +6 C15
 +9 C16 +21 C17 +18 C18 +6 C19 +9 C20 +21 C21 +18 C22 +6 C23 +9 C24 +21 C25 +18 C26 +6 C27 +9 C28 +21 C29
 +18 C30 +6 C31 +9 C32 +21 C33 +18 C34 +6 C35 +9 C36 +21 C37 +19 C38 +6 C39 +8 C40 +21 C41 +19 C42 +6 C43
 +8 C44 +21 C45 +19 C46 +6 C47 +8 C48 +21 C49 +19 C50 <= 200;
Volume: +8 C1 +11 C2 +10 C3 +8 C4 +11 C5 +8 C6 +9 C7 +11 C8 +8 C9 +10 C10 +11 C11 +8 C12 +11 C13 +10 C14 +8 C15
 +11 C16 +8 C17 +9 C18 +11 C19 +8 C20 +10 C21 +11 C22 +8 C23 +11 C24 +10 C25 +8 C26 +11 C27 +8 C28 +9 C29
 +11 C30 +8 C31 +10 C32 +11 C33 +8 C34 +11 C35 +10 C36 +8 C37 +11 C38 +9 C39 +9 C40 +11 C41 +8 C42 +10 C43
 +11 C44 +8 C45 +11 C46 +10 C47 +8 C48 +11 C49 +9 C50 <= 100;

/* Integer definitions */
int C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15,C16,C17,C18,C19,C20,C21,C22,C23,C24,C25,C26,C27,C28,C29,C30,C31,C32,
    C33,C34,C35,C36,C37,C38,C39,C40,C41,C42,C43,C44,C45,C46,C47,C48,C49,C50;
```

Then Mr. X gets the following output: (In here $C1 - C50$ is equivalent to $x_1 - x_{50}$)

```
Value of objective function: 937.00000000


Actual values of the variables:

C6                    10

C19                    1

C50                    1
```

From the output, Mr. X knows to get the most value he will need to take jewelries that are:

$$\text{Index 6 (10 pieces), Index 19 (1 piece), and Index 50 (1 piece)} \qquad \text{(Total of 12 Pieces)}$$

$$Total\ Value = 937\ thousands\ of\ dollars$$

Mr. X notices that there are ten pieces of Jewelry 6 taken if he goes with plan, and from all previous plans, Jewelry 6 is always included. Mr. X wonders why so he calculated the following ratios for Jewelry 6:

$$\frac{value}{weight} = \frac{78}{17} \approx 4.59\ thousands\ of\ dollars/kg$$

$$\frac{value}{volume} = \frac{78}{8} \approx 9.75\ thousands\ of\ dollars/liter$$

Then, Mr. X writes the following code in MATLAB to find the highest $\frac{value}{weight}$ and $\frac{value}{volume}$ ratio:

```
value_list = floor(50 + 30 * cos(1:50));
weight_list = floor(14 + 9 * cos(11 * (1:50) + 2));
volume_list = floor(10 + 2 * cos(4 * (1:50) - 1));

value_weight = value_list ./ weight_list;
value_volume = value_list ./ volume_list;

[v1, i1] = max(value_weight);
[v2, i2] = max(value_volume);
```

It turns out that Jewelry 6 has the highest $\frac{value}{volume}$ ratio and Jewelry 7 has the highest $\frac{value}{weight}$ ratio of 14.4 $thousands\ of\ dollars/kg$. Considering the vehicle's 200 kg weight is numerically bigger than the 100 liters volume constraints, it is reasonable that the higher the $\frac{value}{volume}$ ratio the jewelry has, the higher chance that it will be included with the plan despite Jewelry 6 may not has the highest $\frac{value}{weight}$ ratio. Mr. X finds out that in this case, the total value increased from 840 thousand of dollars to 937 thousand of dollars. The total number of pieces stay unchanged. Mr. X decides to go with this plan to get the most profit from this heist.

## III. Plot Twist:

Later on, Mr. X got arrest during the heist, Zero profit was made. After all those hard works, R.I.P. Mr. X