Problem #1

items = [array of items]
C = size of bins
lowerBound = ceiling((total weight of items) / C)
bins = [array of size of lowerBound initialized with 0]

a)

firstFit(items, C, bins)

```
for item in items                          // iterate through items
    isItemChecked = false                  // check if item is put into bin
    for bin in bins                        // iterate through bins
        if item + bin <= C                 // check if item fits to bin
            bin = bin + item               // add item to bin
            isItemChecked = true
            break
    if not isItemChecked                   // if item can't fit to bin, create a new bin
        bins.append(item)

return bins.size()                         // return size of bins
```

Running time:
It takes n times to check every item, and because at most n bins can be made, it takes n times to iterate bins. Thus, it's O(n^2) where n is the number of items.

firstFitDecreasing(items, C, bins)

```
items.reverseSort()                        // sort items in decreasing order

for item in items
    isItemChecked = false
    for bin in bins
        if item + bin <= C
            bin = bin + item
            isItemChecked = true
            break
    if not isItemChecked
        bins.append(item)

return bins.size()
```

Running time:
O(nlogn) + O(n^2) => O(n^2) where n is the number of items.

bestFit(items, C)

```
bins = []

for item in items
    ti = -1                          // index for bin with the tightest space
    minRoomInBin = c                 // minimum space among bins

    // Find a index with the tightest space in bins
    for i from 0 to bins.length()
        // If there's an index with tighter space
        if bins[i] + item <= c and c - bins[i] <= minRoomInBin
            minRoomInBin = c - bins[i]      // update a minimum space
            ti = i                          // update to a new index

    if ti >= 0
        bins[ti] = bins[ti] + item     // Add a item to the bin with the slightest space
    else
        bins.append(item)              // if item can't fit to the bin, create a new bin

return bins.size()
```

Running time:

It takes n times to check every item, at most n times to find the tightest bin. So, the time efficiency is O(n^2).

c)

How I generated instances

- c is a random capacity of bins (10 <= c <= 20). I set the minimum capacity to 10 for meaningful results.
- Total 30 lists are generated with items of weight from 1 to c.
- Each list has at most 30 items.

Results:

I generated the number of bins and total sum of empty spaces in bins for each bin.

	First-Fit	First-Fit-Decreasing	Best-Fit
Equal to lower bound	0	0	0
Has lowest number of bins	21	30	23
Has lowest spaces of bins	21	30	23
Exclusively performs better	0	7	0

- No algorithm has achieved the lower bound which is the minimum number of bins.
- All three algorithms were a tie in 21 test cases for both numbers and spaces of bins.
- First-Fit-Decreasing and Best-Fit were in a tie in 2 test cases for both numbers and spaces.
- First-Fit-Decreasing exclusively performed better in 7 test cases.

Therefore, I concluded that First-Fit-Decreasing algorithm generally performs better than other algorithms. Performance: First-Fit-Decreasing >= Best-Fit >= First-Fit

Problem #2

Used LINDO for problem #2

(a)

<code>

```
min y1 + y2 + y3 + y4 + y5 + y6
ST
4x11 + 4x12 + 4x13 + 6x14 + 6x15 + 6x16 - 10y1 <= 0
4x21 + 4x22 + 4x23 + 6x24 + 6x25 + 6x26 - 10y2 <= 0
4x31 + 4x32 + 4x33 + 6x34 + 6x35 + 6x36 - 10y3 <= 0
4x41 + 4x42 + 4x43 + 6x44 + 6x45 + 6x46 - 10y4 <= 0
4x51 + 4x52 + 4x53 + 6x54 + 6x55 + 6x56 - 10y5 <= 0
4x61 + 4x62 + 4x63 + 6x64 + 6x65 + 6x66 - 10y6 <= 0
x11 + x21 + x31 + x41 + x51 + x61 = 1
x12 + x22 + x32 + x42 + x52 + x62 = 1
x13 + x23 + x33 + x43 + x53 + x63 = 1
x14 + x24 + x34 + x44 + x54 + x64 = 1
x15 + x25 + x35 + x45 + x55 + x65 = 1
x16 + x26 + x36 + x46 + x56 + x66 = 1
END
INT 6
INT x11
INT x12
INT x13
INT x14
INT x15
INT x16
INT x21
INT x22
INT x23
INT x24
INT x25
INT x26
INT x31
INT x32
INT x33
INT x34
INT x35
INT x36
INT x41
INT x42
INT x43
INT x44
INT x45
```

INT x46

INT x51

INT x52

INT x53

INT x54

INT x55

INT x56

INT x61

INT x62

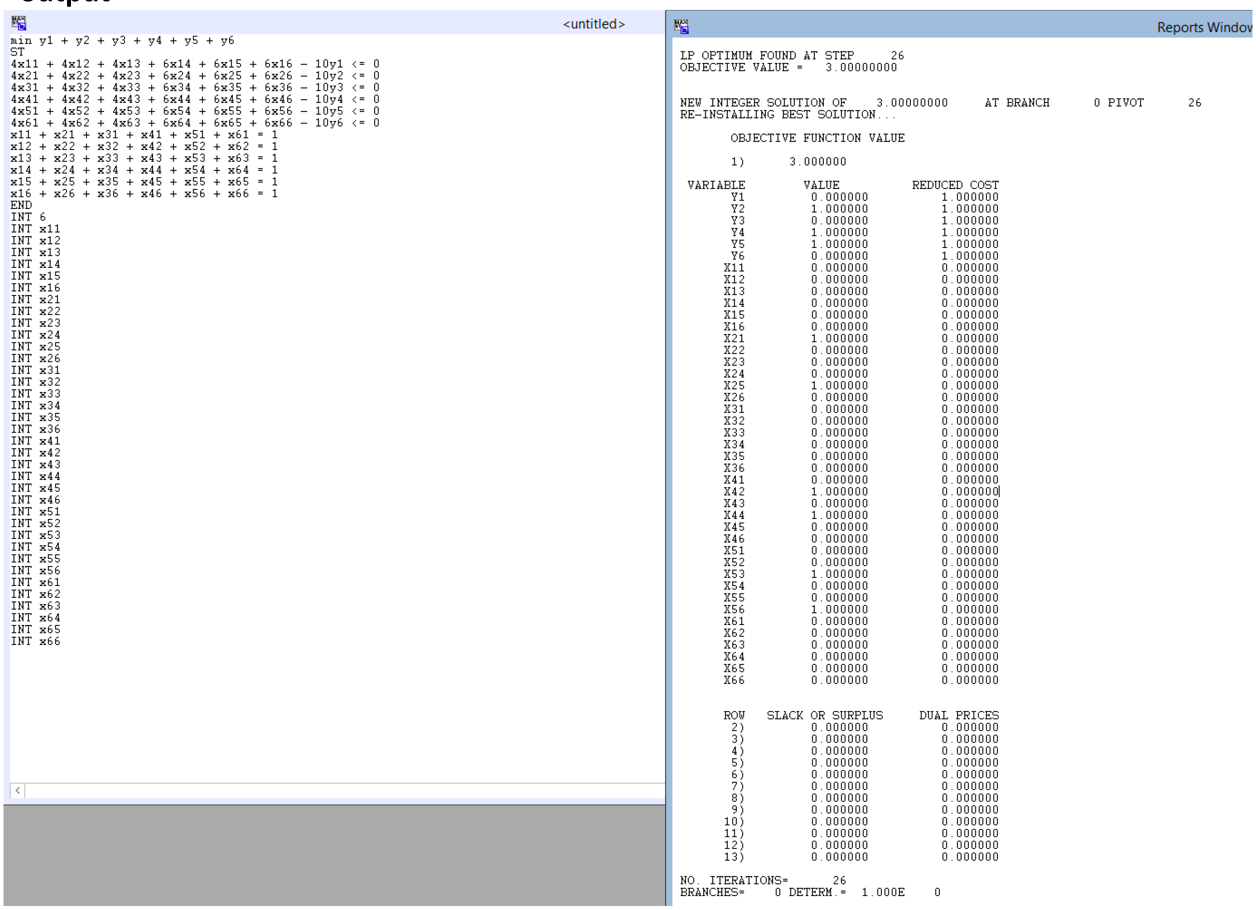
INT x63

INT x64

INT x65

INT x66

<output>



<untitled>

```
min y1 + y2 + y3 + y4 + y5 + y6
ST
4x11 + 4x12 + 4x13 + 6x14 + 6x15 + 6x16 - 10y1 <= 0
4x21 + 4x22 + 4x23 + 6x24 + 6x25 + 6x26 - 10y2 <= 0
4x31 + 4x32 + 4x33 + 6x34 + 6x35 + 6x36 - 10y3 <= 0
4x41 + 4x42 + 4x43 + 6x44 + 6x45 + 6x46 - 10y4 <= 0
4x51 + 4x52 + 4x53 + 6x54 + 6x55 + 6x56 - 10y5 <= 0
4x61 + 4x62 + 4x63 + 6x64 + 6x65 + 6x66 - 10y6 <= 0
x11 + x21 + x31 + x41 + x51 + x61 = 1
x12 + x22 + x32 + x42 + x52 + x62 = 1
x13 + x23 + x33 + x43 + x53 + x63 = 1
x14 + x24 + x34 + x44 + x54 + x64 = 1
x15 + x25 + x35 + x45 + x55 + x65 = 1
x16 + x26 + x36 + x46 + x56 + x66 = 1
END
INT 6
INT x11
INT x12
INT x13
INT x14
INT x15
INT x16
INT x21
INT x22
INT x23
INT x24
INT x25
INT x26
INT x31
INT x32
INT x33
INT x34
INT x35
INT x36
INT x41
INT x42
INT x43
INT x44
INT x45
INT x46
INT x51
INT x52
INT x53
INT x54
INT x55
INT x56
INT x61
INT x62
INT x63
INT x64
INT x65
INT x66
```

Reports Window

```
LP OPTIMUM FOUND AT STEP     26
OBJECTIVE VALUE =    3.00000000


NEW INTEGER SOLUTION OF    3.00000000     AT BRANCH      0 PIVOT      26
RE-INSTALLING BEST SOLUTION...

       OBJECTIVE FUNCTION VALUE

       1)      3.000000

 VARIABLE        VALUE          REDUCED COST
       Y1         0.000000          1.000000
       Y2         1.000000          1.000000
       Y3         0.000000          1.000000
       Y4         1.000000          1.000000
       Y5         1.000000          1.000000
       Y6         0.000000          1.000000
      X11         0.000000          0.000000
      X12         0.000000          0.000000
      X13         0.000000          0.000000
      X14         0.000000          0.000000
      X15         0.000000          0.000000
      X16         0.000000          0.000000
      X21         1.000000          0.000000
      X22         0.000000          0.000000
      X23         0.000000          0.000000
      X24         0.000000          0.000000
      X25         1.000000          0.000000
      X26         0.000000          0.000000
      X31         0.000000          0.000000
      X32         0.000000          0.000000
      X33         0.000000          0.000000
      X34         0.000000          0.000000
      X35         0.000000          0.000000
      X36         0.000000          0.000000
      X41         0.000000          0.000000
      X42         1.000000          0.000000
      X43         0.000000          0.000000
      X44         1.000000          0.000000
      X45         0.000000          0.000000
      X46         0.000000          0.000000
      X51         0.000000          0.000000
      X52         0.000000          0.000000
      X53         1.000000          0.000000
      X54         0.000000          0.000000
      X55         0.000000          0.000000
      X56         1.000000          0.000000
      X61         0.000000          0.000000
      X62         0.000000          0.000000
      X63         0.000000          0.000000
      X64         0.000000          0.000000
      X65         0.000000          0.000000
      X66         0.000000          0.000000


      ROW   SLACK OR SURPLUS     DUAL PRICES
       2)         0.000000          0.000000
       3)         0.000000          0.000000
       4)         0.000000          0.000000
       5)         0.000000          0.000000
       6)         0.000000          0.000000
       7)         0.000000          0.000000
       8)         0.000000          0.000000
       9)         0.000000          0.000000
      10)         0.000000          0.000000
      11)         0.000000          0.000000
      12)         0.000000          0.000000
      13)         0.000000          0.000000

NO. ITERATIONS=      26
BRANCHES=    0 DETERM.=  1.000E    0
```

Total three bins(Y2, Y4, Y5) are filled with items.

Y2 = X21(weight: 4) + X25(weight: 6)

Y4 = X42(weight: 4) + X44(weight: 6)

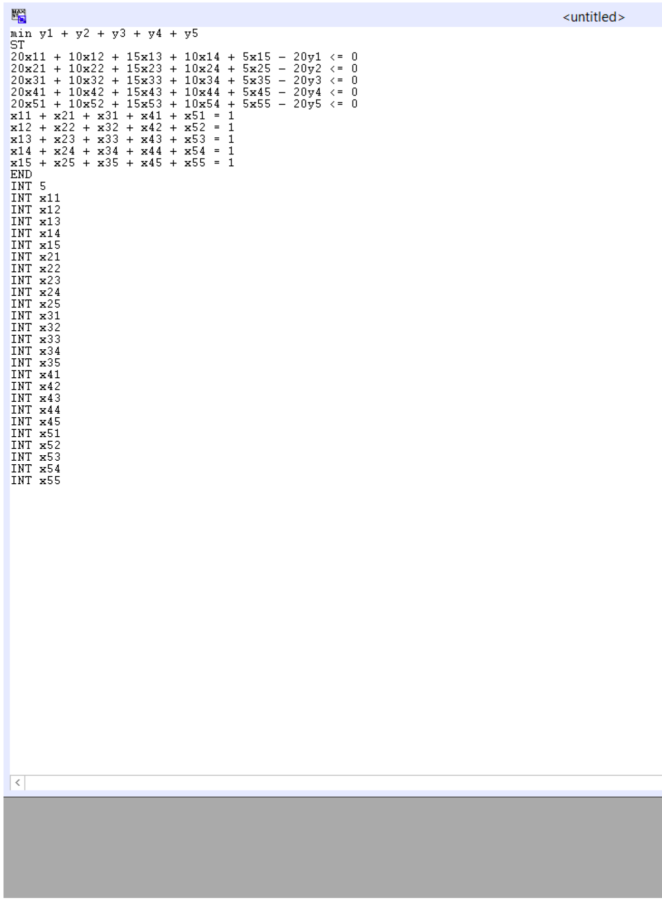
Y5 = X53(weight: 4) + X56(weight: 6)

(b)

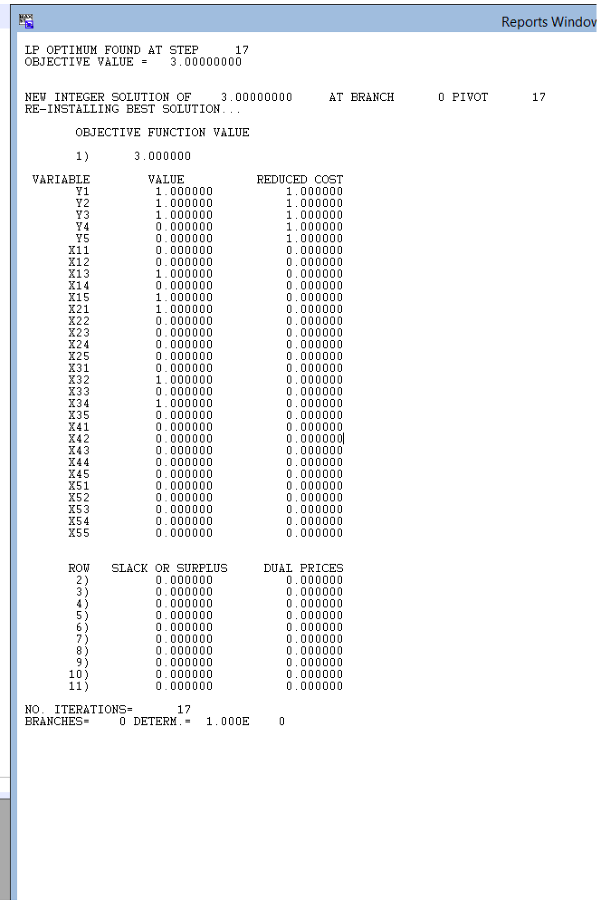
<code>

```
min y1 + y2 + y3 + y4 + y5
ST
20x11 + 10x12 + 15x13 + 10x14 + 5x15 - 20y1 <= 0
20x21 + 10x22 + 15x23 + 10x24 + 5x25 - 20y2 <= 0
20x31 + 10x32 + 15x33 + 10x34 + 5x35 - 20y3 <= 0
20x41 + 10x42 + 15x43 + 10x44 + 5x45 - 20y4 <= 0
20x51 + 10x52 + 15x53 + 10x54 + 5x55 - 20y5 <= 0
x11 + x21 + x31 + x41 + x51 = 1
x12 + x22 + x32 + x42 + x52 = 1
x13 + x23 + x33 + x43 + x53 = 1
x14 + x24 + x34 + x44 + x54 = 1
x15 + x25 + x35 + x45 + x55 = 1
END
INT 5
INT x11
INT x12
INT x13
INT x14
INT x15
INT x21
INT x22
INT x23
INT x24
INT x25
INT x31
INT x32
INT x33
INT x34
INT x35
INT x41
INT x42
INT x43
INT x44
INT x45
INT x51
INT x52
INT x53
INT x54
INT x55
```

<output>



```
<untitled>
min y1 + y2 + y3 + y4 + y5
ST
20x11 + 10x12 + 15x13 + 10x14 + 5x15 - 20y1 <= 0
20x21 + 10x22 + 15x23 + 10x24 + 5x25 - 20y2 <= 0
20x31 + 10x32 + 15x33 + 10x34 + 5x35 - 20y3 <= 0
20x41 + 10x42 + 15x43 + 10x44 + 5x45 - 20y4 <= 0
20x51 + 10x52 + 15x53 + 10x54 + 5x55 - 20y5 <= 0
x11 + x21 + x31 + x41 + x51 = 1
x12 + x22 + x32 + x42 + x52 = 1
x13 + x23 + x33 + x43 + x53 = 1
x14 + x24 + x34 + x44 + x54 = 1
x15 + x25 + x35 + x45 + x55 = 1
END
INT 5
INT x11
INT x12
INT x13
INT x14
INT x15
INT x21
INT x22
INT x23
INT x24
INT x25
INT x31
INT x32
INT x33
INT x34
INT x35
INT x41
INT x42
INT x43
INT x44
INT x45
INT x51
INT x52
INT x53
INT x54
INT x55
```



```
Reports Window
LP OPTIMUM FOUND AT STEP     17
OBJECTIVE VALUE =   3.00000000


NEW INTEGER SOLUTION OF    3.00000000     AT BRANCH      0 PIVOT      17
RE-INSTALLING BEST SOLUTION...

       OBJECTIVE FUNCTION VALUE

       1)      3.000000

 VARIABLE        VALUE          REDUCED COST
       Y1         1.000000          1.000000
       Y2         1.000000          1.000000
       Y3         1.000000          1.000000
       Y4         0.000000          1.000000
       Y5         0.000000          1.000000
      X11         0.000000          0.000000
      X12         0.000000          0.000000
      X13         1.000000          0.000000
      X14         0.000000          0.000000
      X15         1.000000          0.000000
      X21         1.000000          0.000000
      X22         0.000000          0.000000
      X23         0.000000          0.000000
      X24         0.000000          0.000000
      X25         0.000000          0.000000
      X31         0.000000          0.000000
      X32         1.000000          0.000000
      X33         0.000000          0.000000
      X34         1.000000          0.000000
      X35         0.000000          0.000000
      X41         0.000000          0.000000
      X42         0.000000          0.000000
      X43         0.000000          0.000000
      X44         0.000000          0.000000
      X45         0.000000          0.000000
      X51         0.000000          0.000000
      X52         0.000000          0.000000
      X53         0.000000          0.000000
      X54         0.000000          0.000000
      X55         0.000000          0.000000


      ROW   SLACK OR SURPLUS     DUAL PRICES
       2)         0.000000          0.000000
       3)         0.000000          0.000000
       4)         0.000000          0.000000
       5)         0.000000          0.000000
       6)         0.000000          0.000000
       7)         0.000000          0.000000
       8)         0.000000          0.000000
       9)         0.000000          0.000000
      10)         0.000000          0.000000
      11)         0.000000          0.000000

NO. ITERATIONS=      17
BRANCHES=    0 DETERM.=  1.000E    0
```

Total three bins(Y1, Y2, Y3) are filled with items.
Y1 = X13(weight: 15) + X15(weight: 5)
Y2 = X21(weight: 20)
Y3 = X32(weight: 10) + X34(weight: 10)