Calculating Optimum Cost of Managing Different Wastes of Any Municipality of Bangladesh Using C Program

Presented by

Mahmuda Sultana Mimi

Department of Urban and Regional Planning

Bangladesh University of Engineering & Technology

Problem Statement

- Poor waste management is a crucial problem in Bangladesh
- More than half of these wastes are dumped in open spaces illegally without proper treatment, which is threatening for both environment and public health
- So the authority wants to develop a system that can determine quantities of different types of wastes that are to be processed, when the management capacities and unit costs for each types of wastes are provided
- With the help of this system, any municipality can determine the daily quantity of wastes it wants to process and the minimum cost to manage these wastes effectively considering the resource constraints.

Objective

- To develop a C program for any municipality of Bangladesh to determine the total amount of processed wastes based on their type, and the minimum cost to manage these wastes effectively considering the resource constraints.

Data Used

Percentage contribution of waste to different distribution centers:

Type	Waste Type	Recycling Rate (R)	Disposal Rate after Treatment (L)
1	Construction& Demolition	99%	1%
2	Metal	99%	1%
3	Paper/Cardboard	44%	56%
4	Plastics	4%	96%
5	Food	18%	82%
6	Wood	66%	34%
7	Horticultural	73%	27%
8	Ash & Sludge	10%	90%
9	Textile/Leather	4%	96%
10	Used Slag	98%	2%
11	Glass	14%	86%
12	Scrap Tyres	94%	6%
13	Others (stones, ceramic, rubber, etc.)	7%	93%

Collected from:

Waste Statistics and Overall Recycling. (2020). Retrieved from https://www.nea.gov.sg/ourservices/waste-management/waste-statistics-and-overall-recycling

Process Used : Duality Theory

For our problem, we have to

Objective function:

Minimize waste treatment cost (BDT/day),

$$Z = C_1X_1 + C_2X_2 + C_3X_3 + \ldots\ldots\ldots + C_NX_N$$

Constraints:

For landfill disposal, the constraint is:

$$L_1X_1 + L_2X_2 + L_3X_3 + \ldots\ldots\ldots + L_NX_N <= \text{Landfill Capacity}$$

For recycling, the constraint is:

$$R_1X_1 + R_2X_2 + R_3X_3 + \ldots\ldots\ldots + R_NX_N <= \text{Recycling Capacity}$$

Basic Algorithm of Dual Problems

Step 1: Start.
Step 2: Primary matrix formation
Step 3: Transverse matrix
Step 4: Preparation of the matrix for iteration
Step 5: Iteration
Step 6: Checking if iteration is needed again
Step 7: If yes, go to step 4
Step 8: If no, print the answers
Step 9: End.

Step 1

Start

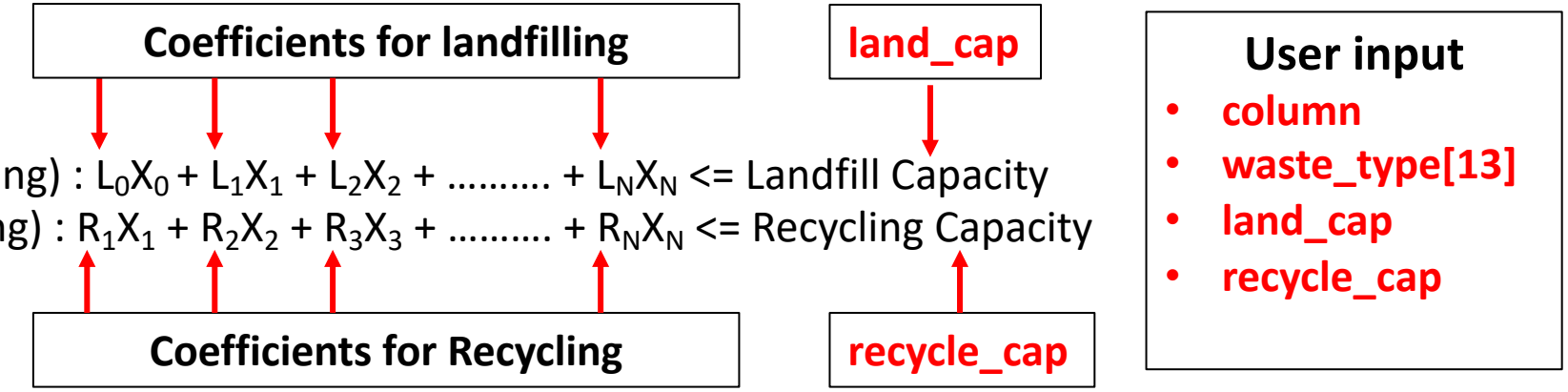
Step 2

Primary matrix formation

N= Total number of types of wastes = 13, Index starts from 0, ends at 12;

Number of type of wastes = Number of variables = Number of columns in the matrix = stored at 'column' variable
Storing the index or type number at: int waste_type[13];

Variables: $X_0, X_1, X_2, \ldots\ldots\ldots\ldots, X_N$ = Total quantity of type wise waste to be processed (unit/ day) **: a[i]**



User input

- **column**
- **waste_type[13]**
- **land_cap**
- **recycle_cap**

L = Percentage of each type of waste to be processed by landfilling disposal which is constant and is fixed in the equation.

R = Percentage of each type of waste to be processed by recycling which is constant and is fixed in the equation.

Step 2

L_0 L_1 L_2 ..... L_{12}

float waste_process_rate[2][13] = {

Coefficients for landfilling → {0.01, 0.01, 0.56, 0.96, 0.82, 0.34, 0.27, 0.90, 0.96, 0.02, 0.86, 0.06, 0.93},

Coefficients for Recycling → {0.99, 0.99, 0.44, 0.04, 0.18, 0.66, 0.73, 0.10, 0.04, 0.98, 0.14, 0.94, 0.07}};

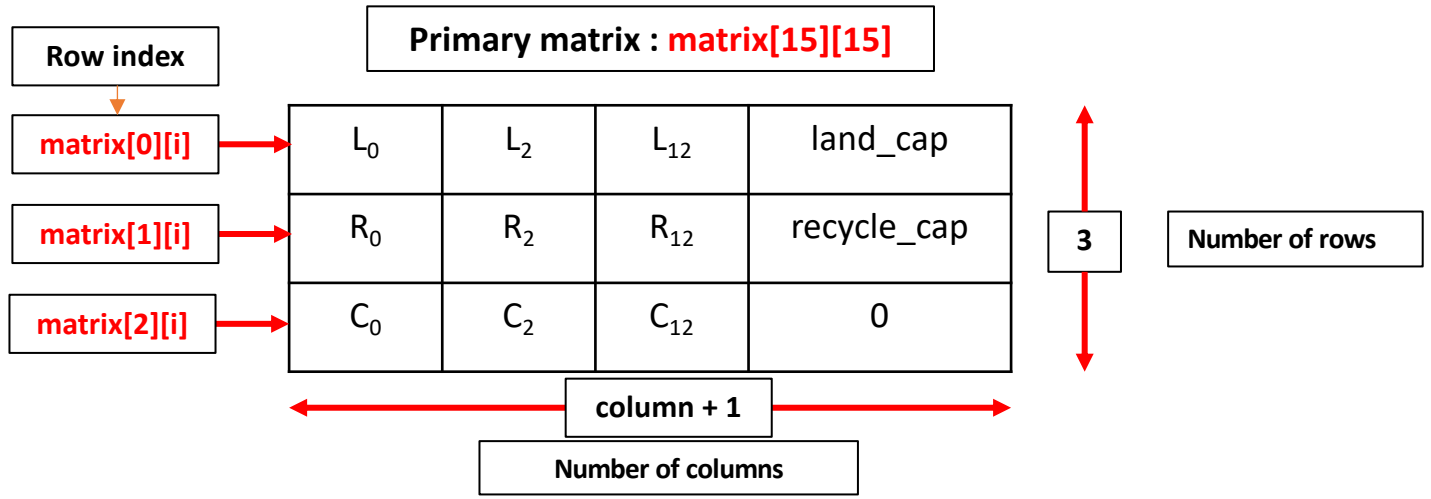
R_0 R_1 R_2 ..... R_{12}

User input: Cost of unit waste processing by waste type (BDT/unit): C_0, C_1, C_2 ....., C_N : stored at **matrix[2][i]**

The total cost of waste processing, (Objective Function) $Z = C_0X_0 + C_1X_1 + C_2X_2 + \ldots\ldots\ldots + C_NX_N$

User input

- **Column = 3**
- **waste_type[0] = 0**
- **waste_type[1] = 2**
- **waste_type[2] = 12**
- **land_cap = 120**
- **recycle_cap= 130**
- **matrix[2][0] = 23 (cost for type 0, C_0)**
- **matrix[2][1] = 34 (cost for type 2, C_2)**
- **matrix[2][2] = 45 (cost for type 12, C_{12})**



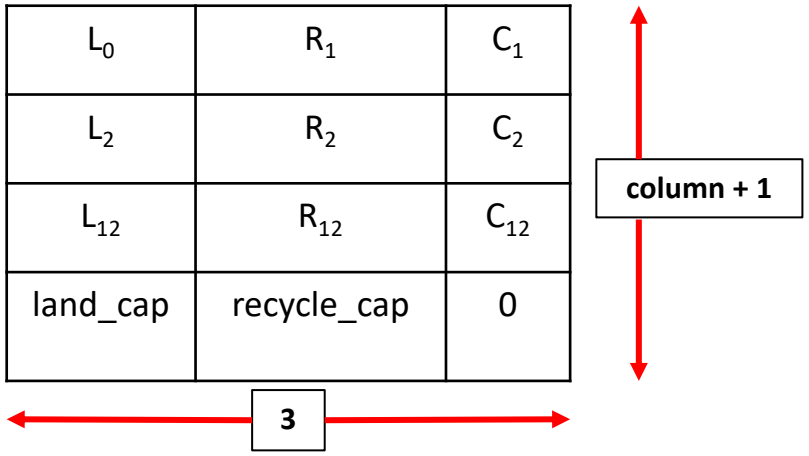
Step 3

Transverse matrix

trans_matrix[15][15]

User input

- **Column = 3**
- **waste_type[0] = 0**
- **waste_type[1] = 2**
- **waste_type[2] = 12**
- **land_cap = 120**
- **recycle_cap= 130**
- **matrix[2][0] = 23 (cost for type 0, C_0)**
- **matrix[2][1] = 34 (cost for type 2, C_2)**
- **matrix[2][2] = 45 (cost for type 12, C_{12})**

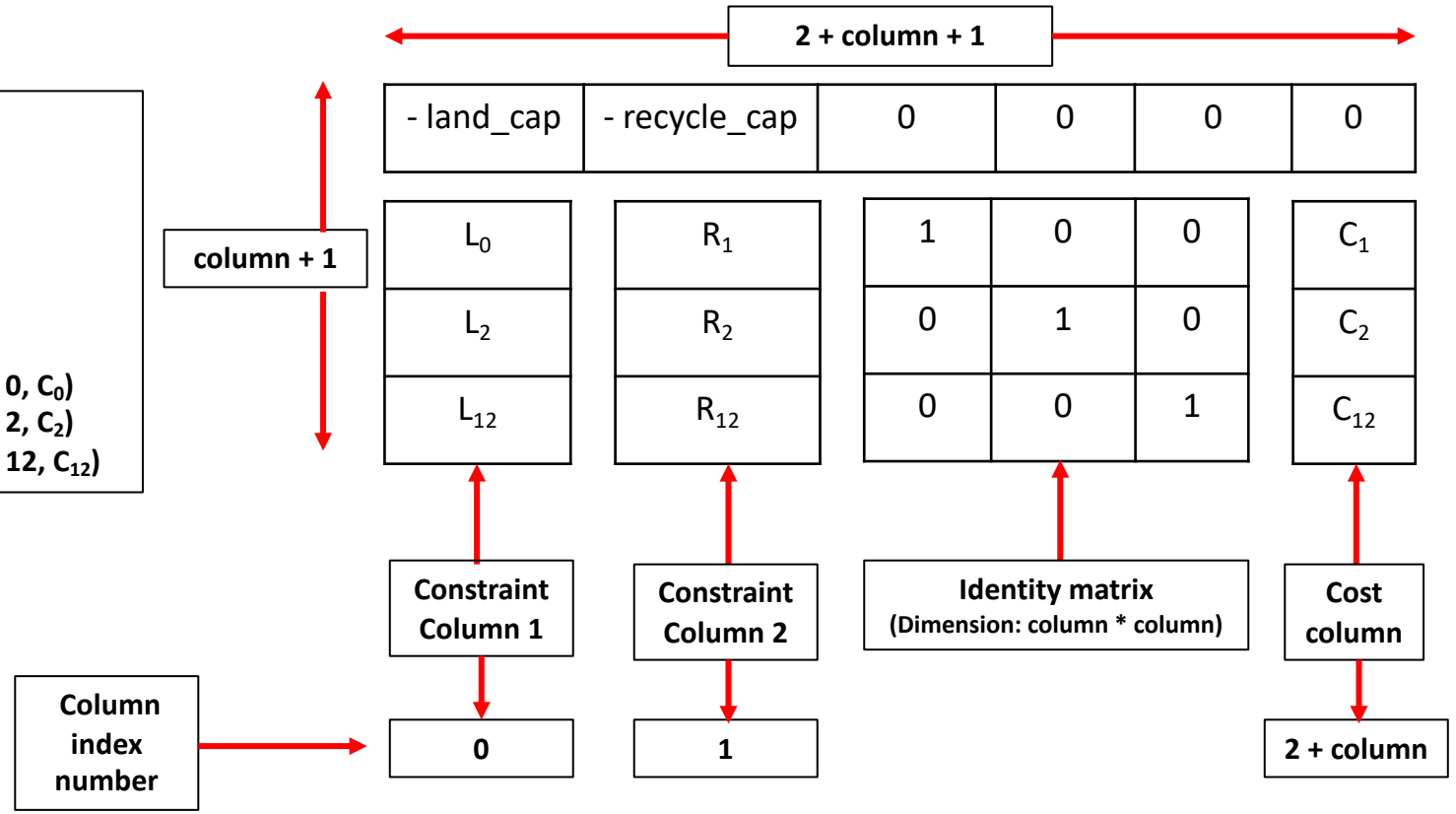


Step 4

Preparation of the matrix for iteration

User input

- Column = 3
- waste_type[0] = 0
- waste_type[1] = 2
- waste_type[2] = 12
- land_cap = 120
- recycle_cap= 130
- matrix[2][0] = 23 (cost for type 0, C_0)
- matrix[2][1] = 34 (cost for type 2, C_2)
- matrix[2][2] = 45 (cost for type 12, C_{12})



Step 5

Iteration

Matrix_new[15][15]

User input

- **Column = 3**
- **waste_type[0] = 0**
- **waste_type[1] = 2**
- **waste_type[2] = 12**
- **land_cap = 120**
- **recycle_cap= 130**
- **matrix[2][0] = 23 (cost for type 0, C_0)**
- **matrix[2][1] = 34 (cost for type 2, C_2)**
- **matrix[2][2] = 45 (cost for type 12, C_{12})**

- 120	-130	0	0	0	0
0.01	0.99	1	0	0	23
0.56	0.44	0	1	0	34
0.93	0.07	0	0	1	45

Constraint Column 1

Constraint Column 2

Identity matrix
(Dimension: 3 * 3)

Cost column

Column index number

0

1

2 + column

Step 5

Iteration

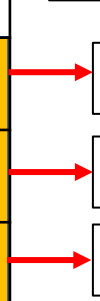
Pivot column:

- 120	-130	0	0	0	0
0.01	0.99	1	0	0	23
0.56	0.44	0	1	0	34
0.93	0.07	0	0	1	45

- 120	-130	0	0	0	0
0.01	0.99	1	0	0	23
0.56	0.44	0	1	0	34
0.93	0.07	0	0	1	45

Pivot row:

- 120	-130	0	0	0	0
0.01	0.99	1	0	0	23
0.56	0.44	0	1	0	34
0.93	0.07	0	0	1	45

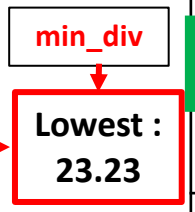


float division[15]

23/ 0.99 = 23.23

34/ 0.44 = 77.27

45/ 0.07 = 642.8



min_div

Lowest : 23.23

- 120	-130	0	0	0	0
0.01	0.99	1	0	0	23
0.56	0.44	0	1	0	34
0.93	0.07	0	0	1	45

Step 5

Iteration

New pivot row:

Matrix_new[15][15]

- 120	-130	0	0	0	0
0.01	0.99	1	0	0	23
0.56	0.44	0	1	0	34
0.93	0.07	0	0	1	45

iteration1[15][15]

0.01/0.99	0.99/0.99	1/0.99	0/0.99	0/0.99	23/0.99

Pivot value = matrix_new[piv_row][piv_col]

New Pivot row = Old pivot row / pivot value

Step 5

Iteration

New other rows:

Matrix_new[15][15]

- 120	-130	0	0	0	0	→	-118.69	0.00	131.31	0.00	0.00	3020.20	New row 0
0.01	0.99	1	0	0	23		0.01	1.00	1.01	0.00	0.00	23.23	New Pivot row
0.56	0.44	0	1	0	34	→	0.56	0.00	-0.44	1.00	0.00	23.78	New row 2
0.93	0.07	0	0	1	45	→	0.93	0.00	-0.07	0.00	1.00	43.37	New row 3

iteration1[15][15]

New row = (Current row) – (Its pivot column coefficient) * (New pivot row)

Step 6

Checking if iteration is needed again

Matrix after first iteration: iteration1[15][15]

Row 0 contains minus value, So iteration is needed again

-118.69	0.00	131.31	0.00	0.00	3020.20
0.01	1.00	1.01	0.00	0.00	23.23
0.56	0.00	-0.44	1.00	0.00	23.78
0.93	0.00	-0.07	0.00	1.00	43.37

Iteration will be continued until row 0 has no minus value

Step 7

If yes, go to step 4

Execution: from step 7 → step 4 → step 5 → step 6

Step 6

Checking if iteration is needed again

Matrix after second iteration: iteration1[15][15]

Row 0 contains no minus value, So further iteration is not needed again

0.00	0.00	36.36	213.64	0.00	8100.00
0.00	1.00	1.02	-0.02	0.00	22.80
1.00	0.00	-0.80	1.80	0.00	43.80
0.00	0.00	0.67	-1.67	1.00	3.60

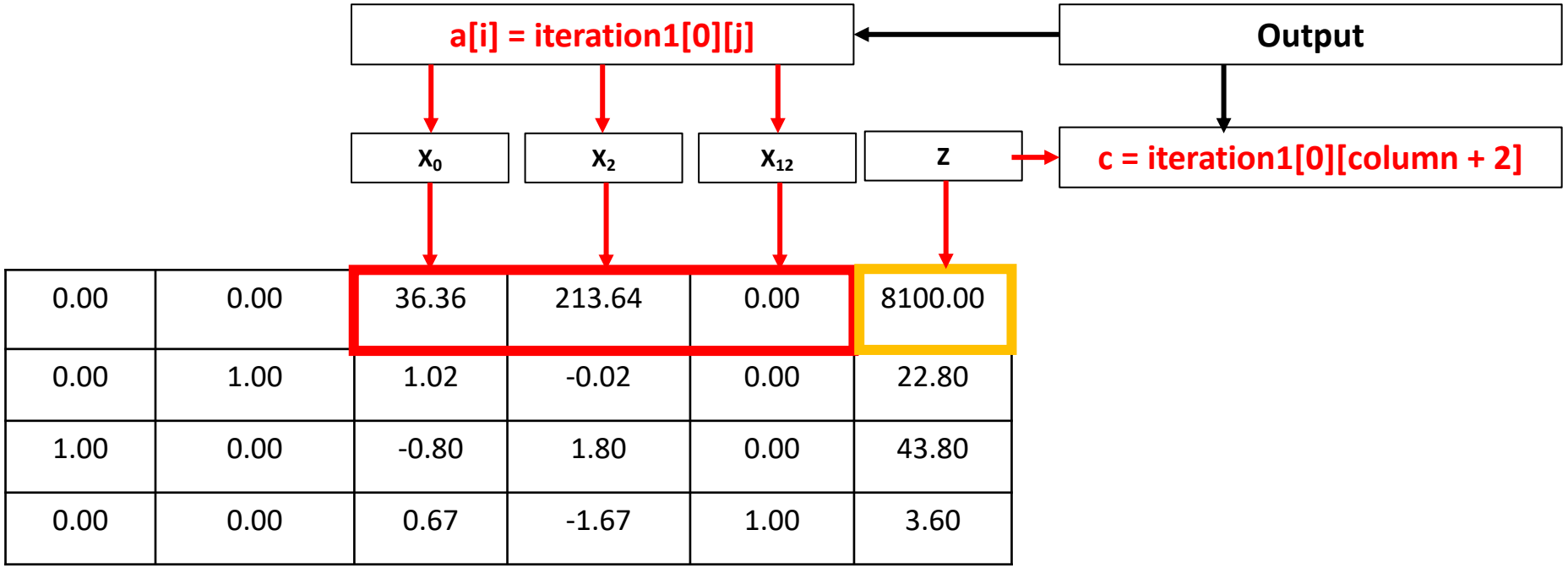
Step 8

If no, print the answers

Step 8

If no, print the answers

Matrix after second iteration: iteration1[15][15]



0.00	0.00	36.36	213.64	0.00	8100.00
0.00	1.00	1.02	-0.02	0.00	22.80
1.00	0.00	-0.80	1.80	0.00	43.80
0.00	0.00	0.67	-1.67	1.00	3.60

Step 9

End

C Program Components

Header files:

- <stdio.h>

Library Function Used:

- main(); printf(); scanf(); fflush(stdin);

Data types used:

- int, float

Operator used:

- **Arithmetic Operator:** Addition (+), Subtraction (-), Multiplication, (*), Division (\), Increment(++)
- **Relational Operator:** >, <, ==
- **Assignment Operator:** =

Escape Sequence: \n, \t

C Program Components

Variable:

- **Local variables of main function:** i, j, row, column, land_cap, recycle_cap, matrix[15][15], tranx_matrix[15][15], waste_process_rate[2][13]
- **Local variables of 'iteration' function:** i, j, min, piv_col, piv_row, matrix_new[15][15], column, waste_type[13], division[15], min_div, iteration1[15][15], a[13], c[2]

Array:

- **One dimensional array:** waste_type[13], a[13], division[15], c[2]
- **Two dimensional array:** waste_process_rate[2][13], matrix[15][15], trans_matrix[15][15], matrix_new[15][15], iteration1[15][15]

C Program Components

Control statement:

- **Decision making/ Selection statement:** if…..else statement
- **Iterative statement:** The for loop, nested for loop
- **Jumping statement:** continue

User defined function:

- **Return type:** void
- **Function name:** iteration
- **Parameters:** matrix_new[15][15], column, waste_type[13])
- **Function call by value**
- **Recursion: Direct recursion**

C Program Components

❖ Recursion: Direct recursion

```
void iteration(float matrix_new[15][15], int column, int waste_type[13]);
int main()
{
    Function_body;
    iteration(matrix_new, column, waste_type);
}
void iteration(float matrix_new[15][15], int column, int waste_type[13])
{
    Function_body;
    if (test == 1)
        {
            iteration(iteration1, column, waste_type);
        }
    else
        {
            //print results
        }

}
```

C Program Components

File management:

- File pointer: file
- Functions used: fopen(), fprint(), fclose()
- File name: Group-6_FINAL_PROJECT_FILE.txt
- File modes used: ‘w’, ‘a’

Group-6_FINAL_PROJECT_FILE.txt

```
.............. User Input ..............

Number of types of waste: 3
Waste types are:
Type: 0
Type: 2
Type: 12

The landfill capacity (tons/day): 120.000000
The recycling capacity (tons/day): 130.000000

The cost of unit waste processing by waste type (BDT/ton):
For waste type 0: 23.000000
For waste type 2: 34.000000
For waste type 12: 45.000000


.............. Program Output ..............

Total quantity of waste to be processed per day by waste types:
Type 0: 36.363647 (tons/day).
Type 2: 213.636353 (tons/day).
Type 12: 0.000000 (tons/day).

Minimum cost required to process wastes: 8100.000000 (BDT/day)
```

Thank You