//PROBLEM 1(WEEK 9)

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n, i, j, k, w;

cin >> n;

int graph[n][n];

string temp;

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

cin >> temp;

if (temp != "INF")

{

graph[i][j] = stoi(temp);

}

else

{

graph[i][j] = 1e8;

}

}

}

for (k = 0; k < n; k++)

{

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

if (graph[i][k] + graph[k][j] < graph[i][j])

{

graph[i][j] = graph[i][k] + graph[k][j];

}

}

}

}

cout << "The shortest path matrix: " << endl;

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

if(graph[i][j] >= 1e8)

cout << "INF";

else

cout << graph[i][j];

cout << " ";

}

cout << endl;

}

return 0;

}

***OUTPUT***

5

0 10 5 5 INF

INF 0 5 5 5

INF INF 0 INF 10

INF 0 10 5 5

0 0 5 5 10

The shortest path matrix:

0 5 5 5 10

5 0 5 5 5

10 10 0 15 10

5 0 5 5 5

0 0 5 5 5

//PROBLEM 2(WEEK 9)

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin >> n;

vector<double> items(n);

vector<double> val(n);

vector<vector<double>> job;

for (int i = 0; i < n; i++)

{

cin >> items[i];

}

for (int i = 0; i < n; i++)

{

cin >> val[i];

job.push\_back({val[i] / items[i], items[i], (double)(i + 1)});

}

double k;

cin >> k;

sort(job.rbegin(), job.rend());

vector<pair<double, double>> ls;

float profit = 0;

for (int i = 0; i < n; i++)

{

if (job[i][1] >= k)

{

profit += k \* job[i][0];

ls.push\_back(make\_pair(k, job[i][2]));

break;

}

else

{

profit += job[i][1] \* job[i][0];

}

ls.push\_back(make\_pair(job[i][1], job[i][2]));

k = k - job[i][1];

}

cout << "Maximum Value : " << profit << endl;

cout << "Item - Weight" << endl;

for (auto it : ls)

cout << it.second << " - " << it.first << endl;

return 0;

}

***OUTPUT***

5

9 7 3 1 0

2 4 5 3 1

10

Maximum Value : -nan

Item - Weight

5 - 0

4 - 1

3 - 3

2 – 6

//PROBLEM 3(WEEK 9)

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin >> n;

vector<int> a(n);

for (int i = 0; i < n; i++)

{

cin >> a[i];

}

priority\_queue<int, vector<int>, greater<int>> minheap;

for (int i = 0; i < n; i++)

{

minheap.push(a[i]);

}

int ans = 0;

while (minheap.size() >1)

{

int e1 = minheap.top();

minheap.pop();

int e2 = minheap.top();

minheap.pop();

ans += e1 + e2;

minheap.push(e1 + e2);

}

cout << ans;

return 0;

}

***OUTPUT***

10

10 30 56 22 100 50 88 10 90 40

1520