**Qn Link :** <https://www.desiqna.in/15782/atlassian-oa-sde-1-freshers-hiring-ctc-75-lac>

**Question :**

* You re given an “N” balloons.
* Each balloons has any of 5 operations( min: 1 opr , max : 5 opr)
  + + by X
  + - by X
  + \* by X
  + / by X
  + Negation ( \* by -1)
* You need to find the maximum value obtained by doing the operations.there is no need to perform the operation.

Tc :

3

3

Opr : + , - , N

4

Opr : \* , N

-5

Opr : \*

Output : 15 (-3 \* -5)

**Observation :**

* A smallest negative value can become the biggest positive value by a negation operation.
* So not only max , we also need min in each step
* From the above example you can know that.
* In one step , we can find find dp[i] value by maxValue of number , min value of the number or nothing do the operation.

Recurrence Relation :

dp\_max [i] = max( dp\_max [i - 1] (opr) value , dp\_min[i - 1] (opr) value , dp[i - 1])

dp\_min [i] = min( dp\_max [i - 1] (opr) value , dp\_min[i - 1] (opr) value , dp[i - 1])

**Code :**

class Solution {

private int min(int a , int b , int c){

if(a < b && a < c){

return a;

}

if(b < c){

return b;

}

return c;

}

private int max(int a , int b , int c){

if(a > b && a > c){

return a;

}

if(b > c){

return b;

}

return c;

}

public int maxValue(int tCase){

int [] dp\_max = new int[n + 1];

int [] dp\_min = new int[n + 1];

dp\_min[1] = dp\_max[1] = 1;

int i = 2;

while(tCase-- != 0){

//Getting the number of ballons

int balloon = in.nextInt();

for(int i = 0 ; i < ballon ; i++){

//Getting operation and value;

char opr = in.next().charAt(0);

int value = in.nextInt();

if(opr == '+'){

dp\_max[i] = max(dp\_max[i - 1] + value , dp\_min[i - 1] + value , dp\_max[i - 1]);

dp\_min[i] = min(dp\_max[i - 1] + value , dp\_min[i - 1] + value , dp\_max[i - 1]);

}else if(opr == '-'){

dp\_max[i] = max(dp\_max[i - 1] - value , dp\_min[i - 1] - value , dp\_max[i - 1]);

dp\_min[i] = min(dp\_max[i - 1] - value , dp\_min[i - 1] - value , dp\_max[i - 1]);

}else if (opr == '\*'){

dp\_max[i] = max(dp\_max[i - 1] \* value , dp\_min[i - 1] \* value , dp\_max[i - 1]);

dp\_min[i] = min(dp\_max[i - 1] \* value , dp\_min[i - 1] \* value , dp\_max[i - 1]);

}else if(opr == '/'){

dp\_max[i] = max(dp\_max[i - 1] / value , dp\_min[i - 1 / value , dp\_max[i - 1]);

dp\_min[i] = min(dp\_max[i - 1] / value , dp\_min[i - 1] / value , dp\_max[i - 1]);

}else{

dp\_max[i] = max(dp\_max[i - 1] \* -1 , dp\_min[i - 1] \* -1 , dp\_max[i - 1]);

dp\_min[i] = min(dp\_max[i - 1] \* -1 , dp\_min[i - 1] \*-1 , dp\_max[i - 1]);

}

}

return dp\_max[n];

}

}

}