A. Genetic Engineering

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You will receive 3 points for solving this problem.

Manao is designing the genetic code for a new type of algae to efficiently produce fuel. Specifically, Manao is focusing on a stretch of DNA that encodes one protein. The stretch of DNA is represented by a string containing only the characters 'A', 'T', 'G' and 'C'.

Manao has determined that if the stretch of DNA contains a maximal sequence of consecutive identical nucleotides that is of even length, then the protein will be nonfunctional. For example, consider a protein described by DNA string "GTTAAAG". It contains four maximal sequences of consecutive identical nucleotides: "G", "TT", "AAA", and "G". The protein is nonfunctional because sequence "TT" has even length.

Manao is trying to obtain a functional protein from the protein he currently has. Manao can insert additional nucleotides into the DNA stretch. Each additional nucleotide is a character from the set {'A', 'T', 'G', 'C'}. Manao wants to determine the minimum number of insertions necessary to make the DNA encode a functional protein.

Input

The input consists of a single line, containing a string s of length n $(1 \leq n \leq 100)$. Each character of s will be from the set {'A', 'T', 'G', 'C'}.

This problem doesn't have subproblems. You will get 3 points for the correct submission.

Output

The program should print on one line a single integer representing the minimum number of 'A', 'T', 'G', 'C' characters that are required to be inserted into the input string in order to make all runs of identical characters have odd length.

Sample test(s)

input

```
GTTAAAG
```

output

```
1
```

input

```
AACCAACCAAAAC
```

output

```
5
```

Note

In the first example, it is sufficient to insert a single nucleotide of any type between the two 'T's in the sequence to restore the functionality of the protein.

B. Word Folding

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You will receive 5 points for solving this problem.

Manao has invented a new operation on strings that is called folding. Each fold happens between a pair of consecutive letters and places the second part of the string above first part, running in the opposite direction and aligned to the position of the fold. Using this operation, Manao converts the string into a structure that has one more level than there were fold operations performed. See the following examples for clarity.

We will denote the positions of folds with '`|`' characters. For example, the word "`ABRACADABRA`" written as "`AB|RACA|DAB|RA`" indicates that it has been folded three times: first, between the leftmost pair of '`B`' and '`R`' letters; second, between '`A`' and '`D`'; and third, between the rightmost pair of '`B`' and '`R`' letters. Here are several examples of folded strings:

```
"ABCDEF|GHIJK" |  "A|BCDEFGHIJK" |  "AB|RACA|DAB|RA" |  "X|XXXXX|X|X|XXXXXX"
               |                 |                   |       XXXXXX
    KJIHG      |   KJIHGFEDCB    |      AR           |       X
   ABCDEF      |            A    |     DAB           |       X
               |                 |     ACAR          |       XXXXX
               |                 |       AB          |           X
```

One last example for "`ABCD|EFGH|IJ|K`":

```
 K
IJ
HGFE
ABCD
```

Manao noticed that each folded string can be viewed as several piles of letters. For instance, in the previous example, there are four piles, which can be read as "`AHI`", "`BGJK`", "`CF`", and "`DE`" from bottom to top. Manao wonders what is the highest pile of identical letters he can build using fold operations on a given word. Note that the pile should not contain gaps and should start at the bottom level. For example, in the rightmost of the four examples above, none of the piles would be considered valid since each of them has gaps, starts above the bottom level, or both.

Input

The input will consist of one line containing a single string of n characters with $1 \leq n \leq 1000$ and no spaces. All characters of the string will be uppercase letters.

This problem doesn't have subproblems. You will get 5 points for the correct submission.

Output

Print a single integer — the size of the largest pile composed of identical characters that can be seen in a valid result of folding operations on the given string.

Sample test(s)

input
```
ABRACADABRA
```
output
```
3
```

input
```
ABBBCBDB
```
output
```
3
```

input
```
AB
```
output
```
1
```

Note

Consider the first example. Manao can create a pile of three '`A`'s using the folding "`AB|RACAD|ABRA`", which results in the following structure:

```
ABRA
DACAR
   AB
```

In the second example, Manao can create a pile of three 'B's using the following folding: "AB|BB|CBDB".

```
CBDB
BB
AB
```

Another way for Manao to create a pile of three 'B's with "ABBBCBDB" is the following folding: "AB|B|BCBDB".

```
 BCBDB
 B
AB
```

In the third example, there are no folds performed and the string is just written in one line.

C1. The Tournament

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of three subproblems: for solving subproblem C1 you will receive 4 points, for solving subproblem C2 you will receive 4 points, and for solving subproblem C3 you will receive 8 points.

Manao decided to pursue a fighter's career. He decided to begin with an ongoing tournament. Before Manao joined, there were n contestants in the tournament, numbered from 1 to n. Each of them had already obtained some amount of tournament points, namely the i-th fighter had p_i points.

Manao is going to engage in a single fight against each contestant. Each of Manao's fights ends in either a win or a loss. A win grants Manao one point, and a loss grants Manao's opponent one point. For each i, Manao estimated the amount of effort e_i he needs to invest to win against the i-th contestant. Losing a fight costs no effort.

After Manao finishes all of his fights, the ranklist will be determined, with 1 being the best rank and $n + 1$ being the worst. The contestants will be ranked in descending order of their tournament points. The contestants with the same number of points as Manao will be ranked better than him if they won the match against him and worse otherwise. The exact mechanism of breaking ties for other fighters is not relevant here.

Manao's objective is to have rank k or better. Determine the minimum total amount of effort he needs to invest in order to fulfill this goal, if it is possible.

Input

The first line contains a pair of integers n and k ($1 \le k \le n + 1$). The i-th of the following n lines contains two integers separated by a single space — p_i and e_i ($0 \le p_i, e_i \le 200000$).

The problem consists of three subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem C1 (4 points), the constraint $1 \le n \le 15$ will hold.
- In subproblem C2 (4 points), the constraint $1 \le n \le 100$ will hold.
- In subproblem C3 (8 points), the constraint $1 \le n \le 200000$ will hold.

Output

Print a single number in a single line — the minimum amount of effort Manao needs to use to rank in the top k. If no amount of effort can earn Manao such a rank, output number -1.

Sample test(s)

input
```
3 2
1 1
1 4
2 2
```
output
```
3
```

input
```
2 1
3 2
4 0
```
output
```
-1
```

input
```
5 2
2 10
2 10
1 1
3 1
3 1
```
output
```
12
```

Note

Consider the first test case. At the time when Manao joins the tournament, there are three fighters. The first of them has 1 tournament point and the victory against him requires 1 unit of effort. The second contestant also has 1 tournament point, but Manao needs 4 units of effort to defeat him. The third contestant has 2 points and victory against him costs Manao 2 units of effort. Manao's goal is top be in top 2. The optimal decision is to win against fighters 1 and 3, after which Manao, fighter 2, and fighter 3 will all have 2 points. Manao will rank better than fighter 3 and worse than fighter 2, thus finishing in second place.

Consider the second test case. Even if Manao wins against both opponents, he will still rank third.

C2. The Tournament

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of three subproblems: for solving subproblem C1 you will receive 4 points, for solving subproblem C2 you will receive 4 points, and for solving subproblem C3 you will receive 8 points.

Manao decided to pursue a fighter's career. He decided to begin with an ongoing tournament. Before Manao joined, there were n contestants in the tournament, numbered from 1 to n. Each of them had already obtained some amount of tournament points, namely the i-th fighter had p_i points.

Manao is going to engage in a single fight against each contestant. Each of Manao's fights ends in either a win or a loss. A win grants Manao one point, and a loss grants Manao's opponent one point. For each i, Manao estimated the amount of effort e_i he needs to invest to win against the i-th contestant. Losing a fight costs no effort.

After Manao finishes all of his fights, the ranklist will be determined, with 1 being the best rank and $n+1$ being the worst. The contestants will be ranked in descending order of their tournament points. The contestants with the same number of points as Manao will be ranked better than him if they won the match against him and worse otherwise. The exact mechanism of breaking ties for other fighters is not relevant here.

Manao's objective is to have rank k or better. Determine the minimum total amount of effort he needs to invest in order to fulfill this goal, if it is possible.

Input

The first line contains a pair of integers n and k ($1 \le k \le n+1$). The i-th of the following n lines contains two integers separated by a single space — p_i and e_i ($0 \le p_i, e_i \le 200000$).

The problem consists of three subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem C1 (4 points), the constraint $1 \le n \le 15$ will hold.
- In subproblem C2 (4 points), the constraint $1 \le n \le 100$ will hold.
- In subproblem C3 (8 points), the constraint $1 \le n \le 200000$ will hold.

Output

Print a single number in a single line — the minimum amount of effort Manao needs to use to rank in the top k. If no amount of effort can earn Manao such a rank, output number -1.

Sample test(s)

input

```
3 2
1 1
1 4
2 2
```

output

```
3
```

input

```
2 1
3 2
4 0
```

output

```
-1
```

input

```
5 2
2 10
2 10
1 1
3 1
3 1
```

output

```
12
```

Note

Consider the first test case. At the time when Manao joins the tournament, there are three fighters. The first of them has 1 tournament point and the victory against him requires 1 unit of effort. The second contestant also has 1 tournament point, but Manao needs 4 units of effort to defeat him. The third contestant has 2 points and victory against him costs Manao 2 units of effort. Manao's goal is top be in top 2. The optimal decision is to win against fighters 1 and 3, after which Manao, fighter 2, and fighter 3 will all have 2 points. Manao will rank better than fighter 3 and worse than fighter 2, thus finishing in second place.

Consider the second test case. Even if Manao wins against both opponents, he will still rank third.

C3. The Tournament

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of three subproblems: for solving subproblem C1 you will receive 4 points, for solving subproblem C2 you will receive 4 points, and for solving subproblem C3 you will receive 8 points.

Manao decided to pursue a fighter's career. He decided to begin with an ongoing tournament. Before Manao joined, there were n contestants in the tournament, numbered from 1 to n. Each of them had already obtained some amount of tournament points, namely the i-th fighter had p_i points.

Manao is going to engage in a single fight against each contestant. Each of Manao's fights ends in either a win or a loss. A win grants Manao one point, and a loss grants Manao's opponent one point. For each i, Manao estimated the amount of effort e_i he needs to invest to win against the i-th contestant. Losing a fight costs no effort.

After Manao finishes all of his fights, the ranklist will be determined, with 1 being the best rank and $n+1$ being the worst. The contestants will be ranked in descending order of their tournament points. The contestants with the same number of points as Manao will be ranked better than him if they won the match against him and worse otherwise. The exact mechanism of breaking ties for other fighters is not relevant here.

Manao's objective is to have rank k or better. Determine the minimum total amount of effort he needs to invest in order to fulfill this goal, if it is possible.

Input

The first line contains a pair of integers n and k ($1 \le k \le n+1$). The i-th of the following n lines contains two integers separated by a single space — p_i and e_i ($0 \le p_i, e_i \le 200000$).

The problem consists of three subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem C1 (4 points), the constraint $1 \le n \le 15$ will hold.
- In subproblem C2 (4 points), the constraint $1 \le n \le 100$ will hold.
- In subproblem C3 (8 points), the constraint $1 \le n \le 200000$ will hold.

Output

Print a single number in a single line — the minimum amount of effort Manao needs to use to rank in the top k. If no amount of effort can earn Manao such a rank, output number -1.

Sample test(s)

input

```
3 2
1 1
1 4
2 2
```

output

```
3
```

input

```
2 1
3 2
4 0
```

output

```
-1
```

input

```
5 2
2 10
2 10
1 1
3 1
3 1
```

output

```
12
```

Note

Consider the first test case. At the time when Manao joins the tournament, there are three fighters. The first of them has 1 tournament point and the victory against him requires 1 unit of effort. The second contestant also has 1 tournament point, but Manao needs 4 units of effort to defeat him. The third contestant has 2 points and victory against him costs Manao 2 units of effort. Manao's goal is top be in top 2. The optimal decision is to win against fighters 1 and 3, after which Manao, fighter 2, and fighter 3 will all have 2 points. Manao will rank better than fighter 3 and worse than fighter 2, thus finishing in second place.

Consider the second test case. Even if Manao wins against both opponents, he will still rank third.

D1. Supercollider

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of two subproblems: for solving subproblem D1 you will receive 3 points, and for solving subproblem D2 you will receive 16 points.

Manao is the chief architect involved in planning a new supercollider. He has to identify a plot of land where the largest possible supercollider can be built. The supercollider he is building requires four-way orthogonal collisions of particles traveling at the same speed, so it will consist of four accelerating chambers and be shaped like a plus sign (i.e., +). Each of the four accelerating chambers must be the same length and must be aligned with the Earth's magnetic field (parallel or orthogonal) to minimize interference.

The accelerating chambers need to be laid down across long flat stretches of land to keep costs under control. Thus, Manao has already commissioned a topographical study that has identified all possible maximal length tracts of land available for building accelerating chambers that are either parallel or orthogonal to the Earth's magnetic field. To build the largest possible supercollider, Manao must identify the largest symmetric plus shape from among these candidate tracts. That is, he must find the two tracts of land that form an axis-aligned plus shape with the largest distance from the center of the plus to the tip of the shortest of the four arms of the plus. Note that the collider need not use the entire length of the tracts identified (see the example in the notes).

Input

The first line of the input will contain two single-space-separated integers n, the number of north-south tracts and m, the number of west-east tracts.

Each of the n lines following the first describes a north-south tract. Each such tract is described by three single-space-separated integers x_i, y_i, l_i representing the vertical line segment from (x_i, y_i) to $(x_i, y_i + l_i)$.

Similarly, after the n lines describing north-south tracts follow m similar lines describing the west-east tracts. Each such tract is described by three single-space-separated integers x_i, y_i, l_i representing the horizontal line segment from (x_i, y_i) to $(x_i + l_i, y_i)$.

All x_i and y_i are between -100000000 and 100000000, inclusive. All l_i are between 1 and 100000000, inclusive. No pair of horizontal segments will touch or intersect, and no pair of vertical segments will touch or intersect.

The problem consists of two subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem D1 (3 points), n and m will be between 1 and 1000, inclusive.
- In subproblem D2 (16 points), n and m will be between 1 and 50000, inclusive.

Output

Print one line containing a single integer, the size of the largest supercollider that can be built on one north-south tract and one west-east tract. The size of the supercollider is defined to be the length of one of the four accelerating chambers. In other words, the size of the resulting supercollider is defined to be the distance from the intersection of the two line segments to the closest endpoint of either of the two segments. If no pair of north-south and west-east tracts intersects, it is not possible to build a supercollider and the program should report a maximum size of zero.

Sample test(s)

input

```
1 2
4 0 9
1 1 8
1 2 7
```

output

```
2
```

Note

Consider the example. There is one vertical line segment from (4, 0) to (4, 9) and two horizontal line segments: from (1, 1) to (9, 1) and from (1, 2) to (8, 2). The largest plus shape that can be found among these segments is formed from the only vertical segment and the second of horizontal segments, and is centered at (4, 2).

The program should output 2 because the closest end point of those segments to the center is (4, 0), which is distance 2 from the center point of (4, 2). The collider will be formed by the line segments from (2, 2) to (6, 2) and from (4, 0) to (4, 4).

D2. Supercollider

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of two subproblems: for solving subproblem D1 you will receive 3 points, and for solving subproblem D2 you will receive 16 points.

Manao is the chief architect involved in planning a new supercollider. He has to identify a plot of land where the largest possible supercollider can be built. The supercollider he is building requires four-way orthogonal collisions of particles traveling at the same speed, so it will consist of four accelerating chambers and be shaped like a plus sign (i.e., +). Each of the four accelerating chambers must be the same length and must be aligned with the Earth's magnetic field (parallel or orthogonal) to minimize interference.

The accelerating chambers need to be laid down across long flat stretches of land to keep costs under control. Thus, Manao has already commissioned a topographical study that has identified all possible maximal length tracts of land available for building accelerating chambers that are either parallel or orthogonal to the Earth's magnetic field. To build the largest possible supercollider, Manao must identify the largest symmetric plus shape from among these candidate tracts. That is, he must find the two tracts of land that form an axis-aligned plus shape with the largest distance from the center of the plus to the tip of the shortest of the four arms of the plus. Note that the collider need not use the entire length of the tracts identified (see the example in the notes).

Input

The first line of the input will contain two single-space-separated integers n, the number of north-south tracts and m, the number of west-east tracts.

Each of the n lines following the first describes a north-south tract. Each such tract is described by three single-space-separated integers x_i, y_i, l_i representing the vertical line segment from (x_i, y_i) to $(x_i, y_i + l_i)$.

Similarly, after the n lines describing north-south tracts follow m similar lines describing the west-east tracts. Each such tract is described by three single-space-separated integers x_i, y_i, l_i representing the horizontal line segment from (x_i, y_i) to $(x_i + l_i, y_i)$.

All x_i and y_i are between -100000000 and 100000000, inclusive. All l_i are between 1 and 100000000, inclusive. No pair of horizontal segments will touch or intersect, and no pair of vertical segments will touch or intersect.

The problem consists of two subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem D1 (3 points), n and m will be between 1 and 1000, inclusive.
- In subproblem D2 (16 points), n and m will be between 1 and 50000, inclusive.

Output

Print one line containing a single integer, the size of the largest supercollider that can be built on one north-south tract and one west-east tract. The size of the supercollider is defined to be the length of one of the four accelerating chambers. In other words, the size of the resulting supercollider is defined to be the distance from the intersection of the two line segments to the closest endpoint of either of the two segments. If no pair of north-south and west-east tracts intersects, it is not possible to build a supercollider and the program should report a maximum size of zero.

Sample test(s)

input

```
1 2
4 0 9
1 1 8
1 2 7
```

output

```
2
```

Note

Consider the example. There is one vertical line segment from (4, 0) to (4, 9) and two horizontal line segments: from (1, 1) to (9, 1) and from (1, 2) to (8, 2). The largest plus shape that can be found among these segments is formed from the only vertical segment and the second of horizontal segments, and is centered at (4, 2).

The program should output 2 because the closest end point of those segments to the center is (4, 0), which is distance 2 from the center point of (4, 2). The collider will be formed by the line segments from (2, 2) to (6, 2) and from (4, 0) to (4, 4).

E1. Three Trees

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of two subproblems: for solving subproblem E1 you will receive 11 points, and for solving subproblem E2 you will receive 13 points.

A tree is an undirected connected graph containing no cycles. The distance between two nodes in an unweighted tree is the minimum number of edges that have to be traversed to get from one node to another.

You are given 3 trees that have to be united into a single tree by adding two edges between these trees. Each of these edges can connect any pair of nodes from two trees. After the trees are connected, the distances between all unordered pairs of nodes in the united tree should be computed. What is the maximum possible value of the sum of these distances?

Input

The first line contains three space-separated integers n_1, n_2, n_3 — the number of vertices in the first, second, and third trees, respectively. The following n_1 - 1 lines describe the first tree. Each of these lines describes an edge in the first tree and contains a pair of integers separated by a single space — the numeric labels of vertices connected by the edge. The following n_2 - 1 lines describe the second tree in the same fashion, and the n_3 - 1 lines after that similarly describe the third tree. The vertices in each tree are numbered with consecutive integers starting with 1.

The problem consists of two subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem E1 (11 points), the number of vertices in each tree will be between 1 and 1000, inclusive.
- In subproblem E2 (13 points), the number of vertices in each tree will be between 1 and 100000, inclusive.

Output

Print a single integer number — the maximum possible sum of distances between all pairs of nodes in the united tree.

Sample test(s)

input
```
2 2 3
1 2
1 2
1 2
2 3
```
output
```
56
```

input
```
5 1 4
1 2
2 5
3 4
4 2
1 2
1 3
1 4
```
output
```
151
```

Note

Consider the first test case. There are two trees composed of two nodes, and one tree with three nodes. The maximum possible answer is obtained if the trees are connected in a single chain of 7 vertices.

In the second test case, a possible choice of new edges to obtain the maximum answer is the following:

- Connect node 3 from the first tree to node 1 from the second tree;
- Connect node 2 from the third tree to node 1 from the second tree.

E2. Three Trees

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of two subproblems: for solving subproblem E1 you will receive 11 points, and for solving subproblem E2 you will receive 13 points.

A tree is an undirected connected graph containing no cycles. The distance between two nodes in an unweighted tree is the minimum number of edges that have to be traversed to get from one node to another.

You are given 3 trees that have to be united into a single tree by adding two edges between these trees. Each of these edges can connect any pair of nodes from two trees. After the trees are connected, the distances between all unordered pairs of nodes in the united tree should be computed. What is the maximum possible value of the sum of these distances?

Input

The first line contains three space-separated integers n_1, n_2, n_3 — the number of vertices in the first, second, and third trees, respectively. The following $n_1 - 1$ lines describe the first tree. Each of these lines describes an edge in the first tree and contains a pair of integers separated by a single space — the numeric labels of vertices connected by the edge. The following $n_2 - 1$ lines describe the second tree in the same fashion, and the $n_3 - 1$ lines after that similarly describe the third tree. The vertices in each tree are numbered with consecutive integers starting with 1.

The problem consists of two subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem E1 (11 points), the number of vertices in each tree will be between 1 and 1000, inclusive.
- In subproblem E2 (13 points), the number of vertices in each tree will be between 1 and 100000, inclusive.

Output

Print a single integer number — the maximum possible sum of distances between all pairs of nodes in the united tree.

Sample test(s)

input

```
2 2 3
1 2
1 2
1 2
2 3
```

output

```
56
```

input

```
5 1 4
1 2
2 5
3 4
4 2
1 2
1 3
1 4
```

output

```
151
```

Note

Consider the first test case. There are two trees composed of two nodes, and one tree with three nodes. The maximum possible answer is obtained if the trees are connected in a single chain of 7 vertices.

In the second test case, a possible choice of new edges to obtain the maximum answer is the following:

- Connect node 3 from the first tree to node 1 from the second tree;
- Connect node 2 from the third tree to node 1 from the second tree.

F1. Stock Trading

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of three subproblems: for solving subproblem F1 you will receive 8 points, for solving subproblem F2 you will receive 15 points, and for solving subproblem F3 you will receive 10 points.

Manao has developed a model to predict the stock price of a company over the next n days and wants to design a profit-maximizing trading algorithm to make use of these predictions. Unfortunately, Manao's trading account has the following restrictions:

- It only allows owning either zero or one shares of stock at a time;
- It only allows buying or selling a share of this stock once per day;
- It allows a maximum of k buy orders over the next n days;

For the purposes of this problem, we define a *trade* to a be the act of buying one share of stock on day i, then holding the stock until some day $j > i$ at which point the share is sold. To restate the above constraints, Manao is permitted to make at most k non-overlapping trades during the course of an n-day trading period for which Manao's model has predictions about the stock price.

Even though these restrictions limit the amount of profit Manao can make compared to what would be achievable with an unlimited number of trades or the ability to hold more than one share at a time, Manao still has the potential to make a lot of money because Manao's model perfectly predicts the daily price of the stock. For example, using this model, Manao could wait until the price is low, then buy one share and hold until the price reaches a high value, then sell for a profit, and repeat this process up to k times until n days have passed.

Nevertheless, Manao is not satisfied by having a merely good trading algorithm, and wants to develop an optimal strategy for trading subject to these constraints. Help Manao achieve this goal by writing a program that will determine when to buy and sell stock to achieve the greatest possible profit during the n-day trading period subject to the above constraints.

Input

The first line contains two integers n and k, separated by a single space, with $1 \le k \le \frac{n}{2}$. The i-th of the following n lines contains a single integer p_i $(0 \le p_i \le 10^{12})$, where p_i represents the price at which someone can either buy or sell one share of stock on day i.

The problem consists of three subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem F1 (8 points), n will be between 1 and 3000, inclusive.
- In subproblem F2 (15 points), n will be between 1 and 100000, inclusive.
- In subproblem F3 (10 points), n will be between 1 and 4000000, inclusive.

Output

For this problem, the program will only report the amount of the optimal profit, rather than a list of trades that can achieve this profit.

Therefore, the program should print one line containing a single integer, the maximum profit Manao can achieve over the next n days with the constraints of starting with no shares on the first day of trading, always owning either zero or one shares of stock, and buying at most k shares over the course of the n-day trading period.

Sample test(s)

input
```
10 2
2
7
3
9
8
7
9
7
1
9
```
output
```
15
```

input
```
10 5
2
7
3
9
8
7
9
7
1
9
```

```
output
```

```
21
```

Note

In the first example, the best trade overall is to buy at a price of 1 on day 9 and sell at a price of 9 on day 10 and the second best trade overall is to buy at a price of 2 on day 1 and sell at a price of 9 on day 4. Since these two trades do not overlap, both can be made and the profit is the sum of the profits of the two trades. Thus the trade strategy looks like this:

```
2    | 7    | 3    | 9    | 8    | 7    | 9    | 7    | 1    | 9
buy  |      |      | sell |      |      |      |      | buy  | sell
```

The total profit is then $(9 - 2) + (9 - 1) = 15$.

In the second example, even though Manao is allowed up to 5 trades there are only 4 profitable trades available. Making a fifth trade would cost Manao money so he only makes the following 4:

```
2    | 7    | 3    | 9    | 8    | 7    | 9    | 7    | 1    | 9
buy  | sell | buy  | sell |      | buy  | sell |      | buy  | sell
```

The total profit is then $(7 - 2) + (9 - 3) + (9 - 7) + (9 - 1) = 21$.

F2. Stock Trading

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of three subproblems: for solving subproblem F1 you will receive 8 points, for solving subproblem F2 you will receive 15 points, and for solving subproblem F3 you will receive 10 points.

Manao has developed a model to predict the stock price of a company over the next n days and wants to design a profit-maximizing trading algorithm to make use of these predictions. Unfortunately, Manao's trading account has the following restrictions:

- It only allows owning either zero or one shares of stock at a time;
- It only allows buying or selling a share of this stock once per day;
- It allows a maximum of k buy orders over the next n days;

For the purposes of this problem, we define a *trade* to a be the act of buying one share of stock on day i, then holding the stock until some day $j > i$ at which point the share is sold. To restate the above constraints, Manao is permitted to make at most k non-overlapping trades during the course of an n-day trading period for which Manao's model has predictions about the stock price.

Even though these restrictions limit the amount of profit Manao can make compared to what would be achievable with an unlimited number of trades or the ability to hold more than one share at a time, Manao still has the potential to make a lot of money because Manao's model perfectly predicts the daily price of the stock. For example, using this model, Manao could wait until the price is low, then buy one share and hold until the price reaches a high value, then sell for a profit, and repeat this process up to k times until n days have passed.

Nevertheless, Manao is not satisfied by having a merely good trading algorithm, and wants to develop an optimal strategy for trading subject to these constraints. Help Manao achieve this goal by writing a program that will determine when to buy and sell stock to achieve the greatest possible profit during the n-day trading period subject to the above constraints.

Input

The first line contains two integers n and k, separated by a single space, with $1 \leq k \leq \frac{n}{2}$. The i-th of the following n lines contains a single integer p_i $(0 \leq p_i \leq 10^{12})$, where p_i represents the price at which someone can either buy or sell one share of stock on day i.

The problem consists of three subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem F1 (8 points), n will be between 1 and 3000, inclusive.
- In subproblem F2 (15 points), n will be between 1 and 100000, inclusive.
- In subproblem F3 (10 points), n will be between 1 and 4000000, inclusive.

Output

For this problem, the program will only report the amount of the optimal profit, rather than a list of trades that can achieve this profit.

Therefore, the program should print one line containing a single integer, the maximum profit Manao can achieve over the next n days with the constraints of starting with no shares on the first day of trading, always owning either zero or one shares of stock, and buying at most k shares over the course of the n-day trading period.

Sample test(s)

input

```
10 2
2
7
3
9
8
7
9
7
1
9
```

output

```
15
```

input

```
10 5
2
7
3
9
8
7
9
7
1
9
```

```
output
21
```

Note

In the first example, the best trade overall is to buy at a price of 1 on day 9 and sell at a price of 9 on day 10 and the second best trade overall is to buy at a price of 2 on day 1 and sell at a price of 9 on day 4. Since these two trades do not overlap, both can be made and the profit is the sum of the profits of the two trades. Thus the trade strategy looks like this:

```
2    | 7    | 3    | 9    | 8    | 7    | 9    | 7    | 1    | 9
buy  |      |      | sell |      |      |      |      | buy  | sell
```

The total profit is then $(9 - 2) + (9 - 1) = 15$.

In the second example, even though Manao is allowed up to 5 trades there are only 4 profitable trades available. Making a fifth trade would cost Manao money so he only makes the following 4:

```
2    | 7    | 3    | 9    | 8    | 7    | 9    | 7    | 1    | 9
buy  | sell | buy  | sell |      | buy  | sell |      | buy  | sell
```

The total profit is then $(7 - 2) + (9 - 3) + (9 - 7) + (9 - 1) = 21$.

F3. Stock Trading

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem consists of three subproblems: for solving subproblem F1 you will receive 8 points, for solving subproblem F2 you will receive 15 points, and for solving subproblem F3 you will receive 10 points.

Manao has developed a model to predict the stock price of a company over the next n days and wants to design a profit-maximizing trading algorithm to make use of these predictions. Unfortunately, Manao's trading account has the following restrictions:

- It only allows owning either zero or one shares of stock at a time;
- It only allows buying or selling a share of this stock once per day;
- It allows a maximum of k buy orders over the next n days;

For the purposes of this problem, we define a *trade* to a be the act of buying one share of stock on day i, then holding the stock until some day $j > i$ at which point the share is sold. To restate the above constraints, Manao is permitted to make at most k non-overlapping trades during the course of an n-day trading period for which Manao's model has predictions about the stock price.

Even though these restrictions limit the amount of profit Manao can make compared to what would be achievable with an unlimited number of trades or the ability to hold more than one share at a time, Manao still has the potential to make a lot of money because Manao's model perfectly predicts the daily price of the stock. For example, using this model, Manao could wait until the price is low, then buy one share and hold until the price reaches a high value, then sell for a profit, and repeat this process up to k times until n days have passed.

Nevertheless, Manao is not satisfied by having a merely good trading algorithm, and wants to develop an optimal strategy for trading subject to these constraints. Help Manao achieve this goal by writing a program that will determine when to buy and sell stock to achieve the greatest possible profit during the n-day trading period subject to the above constraints.

Input

The first line contains two integers n and k, separated by a single space, with $1 \le k \le \frac{n}{2}$. The i-th of the following n lines contains a single integer p_i $(0 \le p_i \le 10^{12})$, where p_i represents the price at which someone can either buy or sell one share of stock on day i.

The problem consists of three subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem F1 (8 points), n will be between 1 and 3000, inclusive.
- In subproblem F2 (15 points), n will be between 1 and 100000, inclusive.
- In subproblem F3 (10 points), n will be between 1 and 4000000, inclusive.

Output

For this problem, the program will only report the amount of the optimal profit, rather than a list of trades that can achieve this profit.

Therefore, the program should print one line containing a single integer, the maximum profit Manao can achieve over the next n days with the constraints of starting with no shares on the first day of trading, always owning either zero or one shares of stock, and buying at most k shares over the course of the n-day trading period.

Sample test(s)

input
```
10 2
2
7
3
9
8
7
9
7
1
9
```
output
```
15
```

input
```
10 5
2
7
3
9
8
7
9
7
1
9
```

```
output
21
```

Note

In the first example, the best trade overall is to buy at a price of 1 on day 9 and sell at a price of 9 on day 10 and the second best trade overall is to buy at a price of 2 on day 1 and sell at a price of 9 on day 4. Since these two trades do not overlap, both can be made and the profit is the sum of the profits of the two trades. Thus the trade strategy looks like this:

```
2    | 7    | 3    | 9    | 8    | 7    | 9    | 7    | 1    | 9
buy  |      |      | sell |      |      |      |      | buy  | sell
```

The total profit is then $(9 - 2) + (9 - 1) = 15$.

In the second example, even though Manao is allowed up to 5 trades there are only 4 profitable trades available. Making a fifth trade would cost Manao money so he only makes the following 4:

```
2    | 7    | 3    | 9    | 8    | 7    | 9    | 7    | 1    | 9
buy  | sell | buy  | sell |      | buy  | sell |      | buy  | sell
```

The total profit is then $(7 - 2) + (9 - 3) + (9 - 7) + (9 - 1) = 21$.