NOI 2007 TASKS OVERVIEW

	Executable file	*Input file*	*Output file*
Task 1: GIFT	GIFT.EXE	GIFT.IN	GIFT.OUT
Task 2: RECT	RECT.EXE	RECT.IN	RECT.OUT
Task 3: CIPHER	CIPHER.EXE	CIPHER.IN	CIPHER.OUT
Task 4: HOLE	HOLE.EXE	HOLE.IN	HOLE.OUT
Task 5: JAWBREAK	JAWBREAK.EXE	JAWBREAK.IN	JAWBREAK.OUT
Task 6: STREET	STREET.EXE	STREET.IN	STREET.OUT

Notes:

1. **Each task will be tested on 5 data sets.**
2. **The maximum execution time for every task is 10 seconds.**
3. **Each data set is worth 20 points.**
4. **For each question, the 5 data sets vary in size, starting from small, and ending with sizes around the limits given in each question.**
5. **Either zero mark or full mark (20 points) is awarded to your answer to each data set. There is no partial credit per data set.**
6. **The task statements are contained on 11 pages following this overview page.**
7. **In the event of tie breaking, a bigger-numbered task is favoured over a smaller-numbered one.**

HAPPY PROGRAMMING!

TASK 1: GIFT

The coach of Jacqueline Yo, Olympic swimmer for Singapore, is concerned about her Butterfly stroke. He records her daily timing in milliseconds (a millisecond is one-thousand of a second) and devises a scheme whereby each time she achieves a timing that is lower than the previous day's timing by at least a certain number of milliseconds, he will reward her with a small encouragement gift.

Given a list of daily timings, determine how many gifts Jacqueline would have received.

Input File: `GIFT.IN`

The first line contains 2 integers n and k, where n $(3 \leq n \leq 100)$ is the number of days, and k $(0 < k \leq 100{,}000)$ the desired improvement (in milliseconds). Whenever Jacqueline's timing reduces by at least k milliseconds over the previous day's timing, she will receive a gift from her coach. The first line is then followed by n lines where each line contains a single integer t $(0 < t \leq 100{,}000)$ which is Jacqueline's daily timing in milliseconds. The n timing records are listed in chronological order.

Example

```
6 100
59420
59410
59310
59290
59470
59350
```

Output File: `GIFT.OUT`

The output file consists of a single integer that indicates the number of gifts Jacqueline would have received.

Example

```
2
```

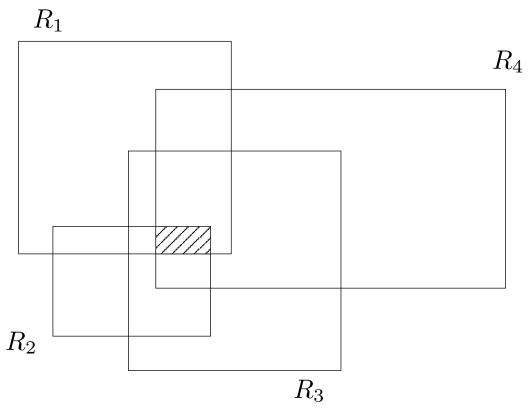
TASK 2: RECT

Given a set of rectangles $\{R_1, R_2, \cdots, R_n\}$, compute the area of their common intersection. i.e.,

$$\text{Area } (R_1 \cap R_2 \cap \cdots \cap R_n)$$

The edges of the rectangles $R_1, R_2, \cdots, R_n$, are either vertical or horizontal lines.

For example, the intersection of 4 rectangles, R_1, R_2, R_3, and R_4, in the following figure is the shaded rectangle.



Input File: `RECT.IN`

The first line specifies the number of rectangles n, where $1 < n < 1{,}000$. Since the sides of the rectangles are parallel to the x and y axes, each rectangle is bounded by the lines $x = x_1$, $x = x_2$, $y = y_1$ and $y = y_2$ and each subsequent line of the input file thus specifies one rectangle in the following format:

```
x1 x2 y1 y2
```

such that $0 \leq x_1 < x_2 \leq 10{,}000$ and $0 \leq y_1 < y_2 \leq 10{,}000$, and x_1, x_2, y_1 and y_2 are integers.

Example 1

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

Example 2

```
3
1 4 1 8
0 2 0 5
10 15 22 35
```

Output File: `RECT.OUT`

The output file contains the area of the intersection of all the rectangles.

Example 1

```
1
```

Example 2

```
0
```

TASK 3: CIPHER

You are the leader of a crack intelligence unit, and today your team intercepted a set of encrypted messages (”ciphertexts”) from “Kojak”, a well known and much feared terrorist leader. It is thought that these ciphertexts contain instructions to his henchmen on which targets to attack next. One of the ciphertexts reads as follows:

```
XLMW MW OSNEO. M EQ LIVIFC SVHIVMRK XLEX EPP QC QIR QYWX
IEX  TITTIVSRM TMDDEW IZIVCHEC. XLMW SVHIV AMPP FI
VITIEPIH SRPC YTSR QC VIXMVIQIRX. PSRK PMZI XLI GLMTQYROW!
```

Your team knows that despite Kojak’s sophistication, he uses a simple Caesar Cipher. In this cipher, letters are shifted by a fixed number of positions P, where P is unknown to you. For example, if letters are shifted by P=2 positions, we get the following “key”:

Plain	A	B	C	D	E	F	G	H	I	J	K	L	M
Cipher	C	D	E	F	G	H	I	J	K	L	M	N	O

Plain	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

So, for example, `HELLO WORLD` is encrypted to `JGNNQ YQTNF`. Kojak encrypts only upper-case letters.

Your team also knows that the words “CHIPMUNKS” and “LIVE” always appear in Kojak’s messages, and that he uses a different key for each message. You can assume that for each message, there is exactly one key that results in the words “CHIPMUNKS” and “LIVE” to appear in the decrypted message. Your task is to write a program to decrypt all of Kojak’s encrypted messages (not just the one shown above) to produce the corresponding plaintext. The fate of your nation is in your hands!

Input File: `CIPHER.IN`

This is a text file containing a single line of ciphertext consisting of at most 1,000 characters.

Example

```
XLMW MW OSNEO. M EQ LIVIFC SVHIVMRK XLEX EPP QC QIR QYWX
IEX  TITTIVSRM TMDDEW IZIVCHEC. XLMW SVHIV AMPP FI
VITIEPIH SRPC YTSR QC VIXMVIQIRX. PSRK PMZI XLI GLMTQYROW!
```

Output File: `CIPHER.OUT`

The plaintext version of the ciphertext. The plaintext version has exactly the same characters as the ciphertext (including newlines, spaces etc), except that uppercase letters are replaced by their decrypted version. Thus, Kojak’s message above leads to the following output:

```
THIS IS KOJAK. I AM HEREBY ORDERING THAT ALL MY MEN MUST
EAT  PEPPERONI PIZZAS EVERYDAY. THIS ORDER WILL BE
REPEALED ONLY UPON MY RETIREMENT. LONG LIVE THE CHIPMUNKS!
```

Hints

You will have to read the ciphertext from the input file. Below, you find how to do this in the three competition languages:

C

```
    FILE* fin = fopen("CIPHER.IN", "r");
    char line[1005];
    fgets(line, sizeof(line), fin);
```

C++

```
    ifstream fin("CIPHER.IN");
    char line[1005];
    fin.getline(line, sizeof(line));
```

Pascal

```
    var
      infile: text;
      str   : array[1..1005] of char;
      ...
    begin
      assign(infile, ’CIPHER.IN’);
      reset(infile);
      readln(infile, str);
```

TASK 4: HOLE

Background: A group of scientists want to monitor a huge forest. They plan to airdrop small sensors to the forest. Due to many unpredictable conditions during airdropping, each sensor will land in a random location in the forest. After all sensors have landed, there will be square regions in the forest that do not contain any sensor. Let us call such a region a *hole*. It is desirable that all holes are small. This can be achieved by airdropping very large number of sensors. On the other hand, those sensors are expensive. Hence, the scientists want to conduct a computer simulation to determine how many sensors should be airdropped, so that the chances of having a large hole are small. To conduct this simulation, a subroutine is required that, given the locations of the sensors, outputs the size of the largest hole. This subroutine has to be very efficient (i.e. fast) since the simulation will be repeated many times with different parameters. You are tasked to write this efficient subroutine.

Problem formulation: Consider an n by n array A of 0 and 1's. We say that there is a hole in A at (x, y) with width d if the value of $A[i, j]$ is 0, where

$$\begin{aligned} i &= x, (x+1), (x+2), \ldots, (x+d-1) \\ j &= y, (y+1), (y+2), \ldots, (y+d-1), \end{aligned}$$

and $(x+d-1) < n, (y+d-1) < n$. That is, all values within the "square" at (x, y) with width d are 0's. Note that the indices of the array start from 0.

Given the array A, we want to find the width of the largest hole.

Example

The following figure shows an array where the entry $A[i, j]$ is at the i-th row and j-th column. The arrow points to the location of $A[3, 7]$.

In this array, there is a hole of width 3 at (0,0), and it is not possible to have a hole of width 4 at (0,0). The largest hole has width 5 and it is at (1,2), which is highlighted in the figure.

	0	1	2	3	4	5	6	7	
0	0	0	0	0	0	1	0	0	
1	0	0	**0**	**0**	**0**	**0**	**0**	0	
2	0	0	**0**	**0**	**0**	**0**	**0**	0	
3	0	1	**0**	**0**	**0**	**0**	**0**	0	$\leftarrow A[3, 7]$
4	0	0	**0**	**0**	**0**	**0**	**0**	0	
5	0	0	**0**	**0**	**0**	**0**	**0**	1	
6	0	0	0	0	1	0	0	0	
7	1	0	0	0	0	0	0	0	

Input File: `HOLE.IN`

The input file represents the input array A. The following is the input file for the array shown above.

```
8
5
3 1
7 0
6 4
0 5
5 7
```

The first line contains a positive integer $n \leq 1{,}024$ which is the width of the array. The second line contains an integer k, which is the number of 1's in the array. Next, it is followed by k more lines which specify the entries with value 1. Each line contains two integers x and y, which indicates that $A[x, y] = 1$.

Since the value of n may be as large as $1{,}024$, your program has to run fast enough to handle arrays of that size.

Output File: `HOLE.OUT`

The output file consists of an integer, which is the width of the largest hole. If there is no hole in the array, the output is 0. The following is the output of our example.

```
5
```

Hints

There are different fast methods that find the largest hole. Below are hints to two different methods.

Hint 1: Consider these 2 values:

d_1:	the width of the largest possible hole at (1,1), and
d_2:	the width of the largest possible hole at (2,2).

What is the relationship between d_1 and d_2? Suppose we know the value of d_2, is there a fast method to compute d_1?

Hint 2: Consider these 2 values:

s_1:	the number of 1's within the square at (0,0) with width 7, and
s_2:	the number of 1's within the square at (0,1) with width 7.

What is the relationship between s_1 and s_2? Suppose we know the value of s_1, is there a fast method to compute s_2?

TASK 5: JAWBREAK

Jawbreaker is a simple game that is often bundled with PDAs and handphones. A player starts with a board filled with coloured balls. The player can select any group of 3 or more adjacent, like-coloured balls to make them disappear. When the selected balls are removed, balls in the same column(s) above the removed balls fall down, leaving empty space at the top. If any columns become empty as a result of this, the remaining columns to the right of the empty columns are compacted to the left, leaving empty space on the right of the board.

A player start with a score of 0. Each move adds to the player's score by the square of the number of balls removed. A player wins Jawbreaker if he/she is able to remove all balls from the board. A win adds a bonus of $1{,}000$ points to the score. The game is over if there are no valid moves remaining on the board.

Adjacent means balls directly above, below or to the left or right. Diagonally touching balls are considered not adjacent.

In this problem, you are given a $n \times n$ Jawbreaker board with m colours. Your program should output the maximum score that can be achieved given the board.

Input File: `JAWBREAK.IN`

The first input line gives the size of the board n and the number of colours m, where ($1 \leq n \leq 8$ and $1 \leq m \leq 4$). The remaining n lines give the initial board, where each of n characters in the line is a number 1 to m, indicating its colour.

Example 1

```
3 3
113
211
112
```

For Example 1, there is only one valid move to remove the group of six 1s, which results in the following board ('-' indicate empty spaces). Note that after the removal of the group of 1s, the second column is empty and is compacted.

```
---
-3-
22-
```

No subsequent moves are possible.

Example 2

```
4 3
3222
3211
2212
3322
```

For Example 2, there are three possible initial moves. Only the removal of the group of three 1s allows the two groups of 2s to collapse into one large group that will result in the maximum score.

Output File: `JAWBREAK.OUT`

Output the maximum score that can be achieved from the initial board.

Example 1

```
36
```

Example 2

```
1106
```

TASK 6: STREET

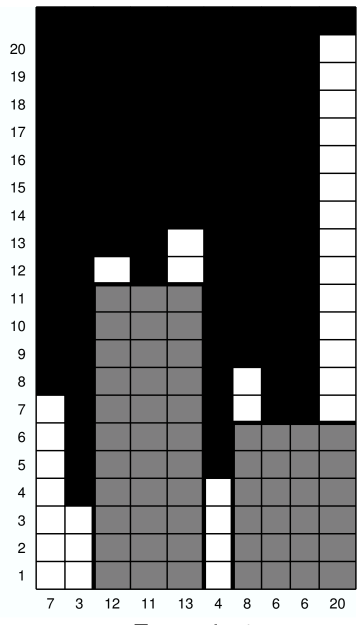
There are n lots on one side of a street (where $n \leq 500$). We would like to erect at most k apartment buildings on these lots. Each building must occupy an interval of at most t consecutive lots. Moreover, each lot i has a height restriction $r[i]$ (where $r[i] \leq 100$). A building cannot exceed any of the height restriction of any lot on which it is built (that is, the maximal height of the building that can be erected on lot i to j is $H = min\{r[i], r[i+1], ..., r[j]\}$). Hence, the maximum usable facade space of the building is $H \times (j - i + 1)$. We would like to have a program to select at most k non-overlapping intervals to erect the buildings such that the total usable facade space is maximized.

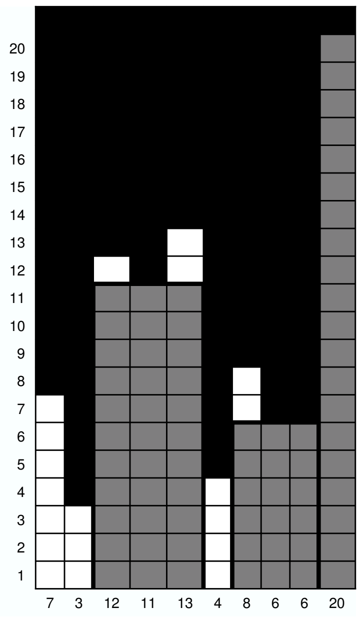
Example 1

Consider a street of length 10. The height restriction of each lot is as follows.

```
7, 3, 12, 11, 13, 4, 8, 6, 6, 20
```

Suppose we would like to erect at most $k = 2$ buildings and each building occupies at most $t = 4$ lots. Then, to maximize the total usable facade space, we choose two intervals $r[3..5] = (12, 11, 13)$ and $r[7..10] = (8, 6, 6, 20)$ (see "Example 1" in the figure below). The maximum usable facade space is $3 * \min\{12, 11, 13\} + 4 * \min\{8, 6, 6, 20\} = 57$.


Example 1


Example 2

Example 2

Suppose we would like to erect at most $k = 3$ buildings on the same street with the same height restrictions as in Example 1, and each building occupies at most $t = 4$ lots. Then, to maximize the total usable facade space, we choose three intervals $r[3..5] = (12, 11, 13)$, $r[7..9] = (8, 6, 6)$ and $r[10..10] = (20)$ (see "Example 2" in the figure above). The maximum usable facade space is $3 * \min\{12, 11, 13\} + 3 * \min\{8, 6, 6\} + 1 * 20 = 71$.

Input File: `STREET.IN`

The input file STREET.IN is as follows. The first line contains three integers n, k, and t separated by a space character, where $1 \leq n \leq 500, 1 \leq k \leq n$, and $1 \leq t \leq n$. The rest of the n lines contain n positive integers representing the height restriction for the n lots. For Example 1, the input file looks like:

```
10 2 4
7
3
12
11
13
4
8
6
6
20
```

Output File: `STREET.OUT`

The output file STREET.OUT contains an integer which is the maximum usable facade space. For the above example, the output file looks like:

```
57
```