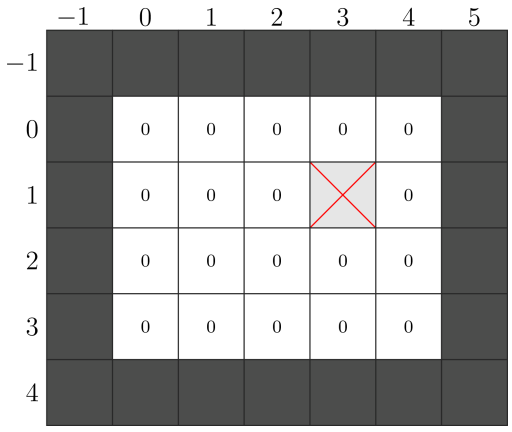
مسابقة الروبوت

سيقيم الباحثون في جامعة سيجيد مسابقة برمجة روبوتات. قرر سهيل أن يشارك في هذه المسابقة. الهدف هو برمجة *Pulibot*، تقديراً بالكلب الهنغاري الشهير من نوع *Puli*. سيتم اختبار *Pulibot* ضمن متاهة على شكل شبكة من $(H+2) \times (W+2)$ خلية. أسطر الشبكة مرقمة من -1 إلى H من الشمال إلى الجنوب، والأعمدة مرقمة من -1 إلى W من الغرب إلى الشرق. نشير إلى الخلية التي تقع في السطر r والعمود c من الشبكة (حيث $-1 \leq r \leq H$, $-1 \leq c \leq W$) بالخلية (r, c).

لتكن الخلية (r, c) حيث $0 \leq r < H$ و $0 \leq c < W$. يوجد 4 خلايا **مجاورة** للخلية (r, c):

- نشير إلى الخلية $(r, c-1)$ بـ **غرب** الخلية (r, c);
- نشير إلى الخلية $(r+1, c)$ بـ **جنوب** الخلية (r, c);
- انشير إلى لخلية $(r, c+1)$ بـ **شرق** الخلية (r, c);
- نشير إلى الخلية $(r-1, c)$ بـ **شمال** الخلية (r, c).

نقول عن خلية (r, c) أنها خلية **حدية** في المتاهة إذا كان $r = -1$ أو $r = H$، أو $c = -1$ أو $c = W$. بقية الخلايا في المتاهة ستكون إما خلية **عقبة** أو خلية **فارغة**. بالإضافة إلى أن كل خلية فارغة لها **لون** معين، يتمثل بعدد صحيح بين 0 و Z_{MAX}، ضمناً. بدايةً، سيكون لون الخلايا الفارغة 0. على سبيل المثال، لتكن المتاهة ذات $H = 4$ و $W = 5$، تحتوي على عقبة وحيدة في الخلية $(1, 3)$:



تم الإشارة إلى العقبة بإشارة ضرب، والخلايا الحدية مظللة. بينما الأعداد المكتوبة ضمن الخلايا تشير إلى اللون الخاص بكل خلية..

نعرف **المسار** ذو الطول ℓ $(\ell > 0)$ من الخلية (r_0, c_0) إلى الخلية (r_ℓ, c_ℓ) بأنه سلسلة من الخلايا *الفارغة* المختلفة $(r_0, c_0), (r_1, c_1), \ldots, (r_\ell, c_\ell)$ التي تحقق من أجل كل i $(0 \leq i < \ell)$ الخلايا (r_i, c_i) و (r_{i+1}, c_{i+1}) تكون متجاورة. لاحظ أن المسار ذو الطول ℓ يحتوي على $\ell + 1$ خلية تماماً.

خلال المسابقة، سيقوم الباحثون بإعداد متاهة بحيث يكون هناك مسار واحد على الأقل من الخلية $(0, 0)$ إلى الخلية $(H-1, W-1)$. لاحظ أن ذلك يضمن أن الخليتين $(0, 0)$ و $(H-1, W-1)$ هما خليتنا فارغتان دائماً.

لا يعلم سهيل أي من الخلايا ضمن المتاهة ستكون خالية وأي منها ستحوي على عقبة.

مهمتك هي مساعدة سهيل على كتابة برنامج للروبوت بحيث يكون قادراً على إيجاد *أقصر طريق* (أي الطريق ذو الطول الأصغري) من الخلية $(0,0)$ إلى الخلية $(H-1,W-1)$ بغض النظر عن المتاهة التي سيقوم الباحثون بإعدادها سنحدد فيما يلي مواصفات الروبوت وما هي قواعد المسابقة

انتبه إلى أنه في نهاية نص المسألة هناك شرح عن أداة مساعدة يمكنك استخدامها مع المصحح التجريبي لمشاهدة حركة الروبوت بشكل مرئي.

مواصفات الروبوت

سنعرف **حالة** الخلية (r,c) من أجل $-1 \leq r \leq H$ و $-1 \leq c \leq W$ بأنها عدد صحيح بحيث:

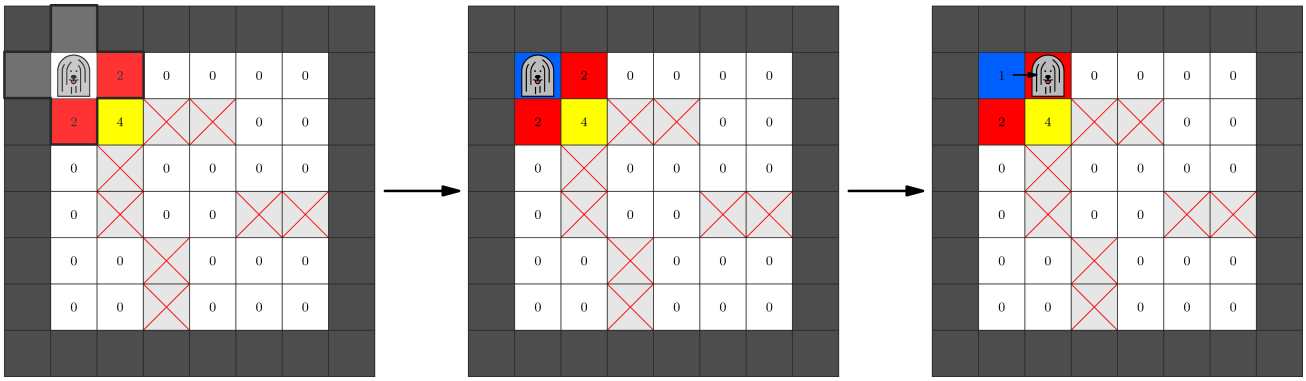
- إذا كانت الخلية (r,c) هي خلية حدية تكون حالتها -2;
- إذا كانت الخلية (r,c) هي خلية عقبة تكون حالتها -1;
- إذا كانت الخلية (r,c) هي خلية فارغة عندها تكون حالتها هي رقم يعبر عن لون هذه الخلية.

سيتم تنفيذ البرنامج الخاص بالروبوت على شكل سلسلة من الخطوات، في كل خطوة يقوم الروبوت بتحديد حالة الخلايا المجاورة لموقعه الحالي وبناء على هذه الحالة يقوم بتنفيذ أمر معين. هذا الأمر الذي يقوم الروبوت بتنفيذه يتم تحديده من خلال الحالة التي قام الروبوت بتحديدها للخلايا المجاورة وفي ما يلي وصف أكثر دقة لذلك:

لنفرض أنه في بداية الخطوة الحالية كان الروبوت واقفاً عنند الخلية (r,c) ، والتي يجب أن تكون خلية فارغة، سيقوم الروبوت بالخطوات كالتالي:

1. يقوم الروبوت بتحديد **صفوفة الحالات** الحالية، وهي المصوفة $S = [S[0], S[1], S[2], S[3], S[4]]$ ، المكونة من حالات الخلية , (r,c) وكل الخلايا المجاورة:
   - $S[0]$ هي حالة الخلية (r,c).
   - $S[1]$ هي حالة الخلية إلى الغرب.
   - $S[2]$ هي حالة الخلية إلى الجنوب.
   - $S[3]$ هي حالة الخلية إلى الشرق.
   - $S[4]$ هي حالة الخلية إلى الشمال.
2. بعد ذلك يقوم الروبوت بتحديد **الأمر** (Z,A) المتوافق مع مصفوفة الحالات التي تم إيجادها. أي أنه لكل مصفوفة حالات يوجد أمر مرتبط بهذه المصفوفة.
3. أخيراً يقوم الروبوت بتنفيذ هذا الأمر كالتالي: يقوم بتغيير لون الخلية (r,c) إلى اللون Z ومن ثم يقوم بالتحرك بناء على قيمة A, والتي يمكن أن تكون واحدة من التصرفات التالية:
   - *البقاء* في الخلية (r,c);
   - *التحرك* إلى واحدة من الخلايا 4 المجاورة;
   - *إنهاء البرنامج*.

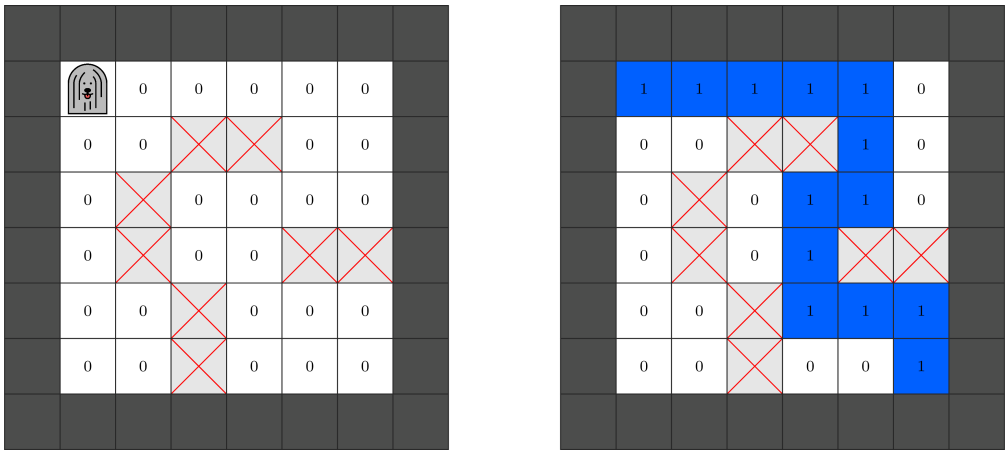
كمثال على ذلك، لنفرض السيناريو المعروض على يسار الشكل التالي: الروبوت هو حالياً في الخلية $(0,0)$ التي لونها 0. يقوم الروبوت بتحديد قيم مصفوفة الحالات ليجدها $S = [0,-2,2,2,-2]$. يقوم الروبوت بالعودة إلى برنامجه الذي تم تحديده له وتحديد الأمر المرتبط مع مصفوفة الحالات التي وجدها، ممكن أن يكون هذا الأمر مثلاً هو أن يغير لون الخلية الحالية إلى $Z = 1$ ومن ثم يتحرك إلى الشرق كما هو معروض في منتصف ويمين الشكل.



قواعد مسابقة الروبوت

- في البداية يتم وضع الروبوت في الخلية $(0, 0)$ ويتم تشغيل برنامجه.
- من غير المسموح ان يتحرك الروبوت إلى خلية غير خالية.
- يجب على برنامج الروبوت أن ينتهي بعد $500\,000$ خطوة على الأكثر.
- بعد أن ينتهي برنامج الروبوت يجب أن تكون الخلايا الفارغة ضمن المتاهة ملونة كما يلي:
  - هناك مسار أصغري من $(0, 0)$ إلى $(H - 1, W - 1)$ بحيث أن لون كل خلية تقع على هذا المسار هو 1.
  - لون كل الخلايا الفارغة الأخرى يجب أن يكون 0.
- يمكن للروبوت أن ينهي برنامجه عندما يكون واقفاً عند أي خلية فارغة أي أن المهمة هي تلوين الخلايا التي تمثل أحد الطرق الأصغرية وليس إيصال الروبوت.

كمثال على ذلك، يعرض الشكل التالي مثالاً على متاهة ممكنة بحيث $H = W = 6$. الوضع الابتدائي معروض على اليسار وأحد احتمالات التلوين المطلوبة للخلايا الفارغة معروض على اليمين والذي يجب أن يتحقق بعد أن ينهي الروبوت برنامجه.



تفاصيل البرمجة

يجب عليك برمجة الإجرائية التالية:

```
void program_pulibot()
```

- هذه الإجرائية يجب أن تقوم بتوليد برنامج الروبوت، وهو عبارة عن مجموعة من القواعد تربط كل مصفوفة حالات مع أمر معين، يجب على هذا البرنامج أن يعمل بشكل صحيح من أجل أي متاهة تحقق شروط المسالة

ومن أجل أي قيمة لـ H و W
- سيتم طلب هذه الإجرائية مرة واحدة تماماً من أجل كل حالة اختبار

لتقوم هذه الإجرائية بتحديد قواعد عمل الروبوت يمكنها أن تقوم بعدة استدعاءات للتابع التالي:

```
void set_instruction(int[] S, int Z, char A)
```

- S: مصوفة طولها 5 تصف مصفوفة الحالات.
- Z: عدد صحيح غير سالب يحدد اللون .
- A: حرف وحيد يمثل التصرف الذي يجب على الروبوت أن يقوم به كالتالي:
  - `H`: ابق مكانك;
  - `W`: تحرك إلى الغرب;
  - `S`: تحرك إلى الجنوب;
  - `E`: تحرك إلى الشرق;
  - `N`: تحرك إلى الشمال;
  - `T`: قم بإنهاء البرنامج.
- طلب هذا الإجراء يعلم الروبوت أنه عندما تجد أن مصفوفة الحالات هي S يجب عليك تنفيذ الأمر (Z, A).

إذا قمت بطلب هذه الإجرائية أكثر من مرة مع نفس الحالة S ستحصل على النتيجة التالية `Output isn't correct`.

ليس من المطلوب استدعاء الإجرائية السابقة من أجل أي مصفوفة حالات ممكنة S. ولكن إذا صادف الروبوت لاحقاً حالة لم تقم بإعطائه أمر خاص بها ستحصل على النتيجة التالية: `Output isn't correct`.

بعد أن ينتهي الإجراء `program_pulibot` سيقوم المصحح بتشغيل البرنامج على واحدة أو اكثر من المتاهات، الوقت اللازم لتشغيل برنامج الروبوت على المتاهات *لا* يتم احتسابه من ضمن الحدود الزمنية لحلك. المصحح *ليس* متكيفاً أي ان مجموعة المتهاهات محددة سابقاً من أجل كل حالة اختبار ولن تتغير بحسب برنامج الروبوت الذي قمت بوضعه.

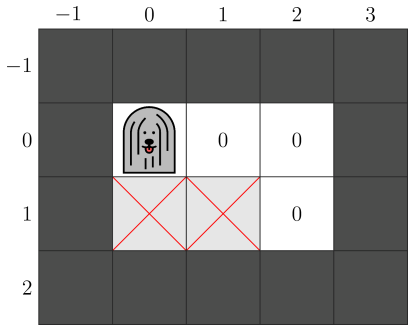
إذا قام الروبوت بمخالفة أي قاعدة من قواعد مسابقة الروبوت قبل أن ينهي برنامجه ستحصل على النتيجة التالية: `Output isn't correct`.

مثال

يمكن للإجراء `program_pulibot` أن يستدعي الإجرائية `set_instruction` كالتالي:

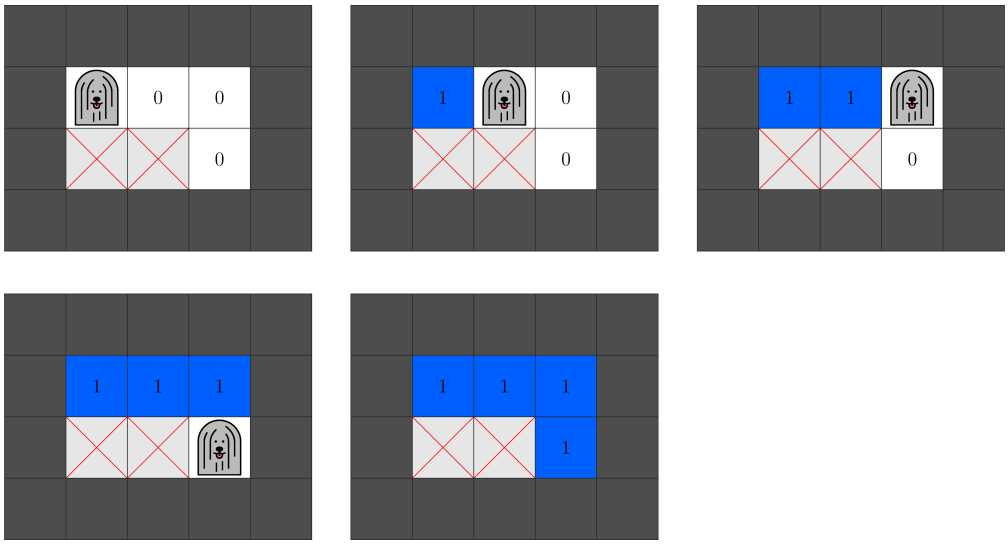
الأمر المرتبط بمصفوفة الحالات S	الاستدعاء
Set color to 1 and move east	`set_instruction([0, -2, -1, 0, -2], 1, E)`
Set color to 1 and move east	`set_instruction([0, 1, -1, 0, -2], 1, E)`
Set color to 1 and move south	`set_instruction([0, 1, 0, -2, -2], 1, S)`
Set color to 1 and terminate program	`set_instruction([0, -1, -2, -2, 1], 1, T)`

ليكون لدينا المتاهة التالية بحيث $H = 2$ و $W = 3$, التي سيتم تجريب البرنامج السابق عليها والمعروضة في الشكل التالي:



من أجل هذه المتاهة بالذات سيعمل البرنامج الخاص بالروبوت بأربع خطوات. مصفوفات الحالات التي سيقوم الروبوت باكتشافها والأوامر التي سيقوم بتنفيذها تتوافق تماماً مع الاستدعاءات الأربعة للإجرائية `set_instruction` التي قمنا بها في الأعلى بالترتيب ولاحظ أن آخر أمر من هذه الأربعة يقوم بإنهاء البرنامج

يعرض الشكل التالي حالة المتاهة بعد كل واحدة من الخطوات الأربعة وحالة المتاهة النهائية بعد انهاء برنامج الروبوت



على كل حال لاحظ أن هذا البرنامج المعروض في المثال والمؤلف من أربع خطوات يمكن أن لا يعطي أقصر طريق في متاهات أخرى غير هذه المتاهة لذلك لو قمت بإرسال هذا الحل ستحصل على النتيجة `Output isn't correct`.

القيود

$Z_{MAX} = 19$. أي أن الروبوت يمكنه تلوين الخلايا من 0 حتى 19, ضمناً.

من أجل كل متاهة سيتم تجريب برنامج الروبوت عليها:

- $2 \leq H, W \leq 15$
- يوجد مسار واحد على الأقل من الخلية $(0, 0)$ إلى الخلية $(H - 1, W - 1)$.

المسائل الجزئية

1. (6 نقاط) لا يوجد خلايا تحوي عقبات ضمن المتاهة.
2. (10 نقطة) $H = 2$
3. (18 نقطة) يوجد تماماً مسار واحد بين أي خليتين خاليتين.
4. (20 نقطة) أي مسار أصغري من الخلية $(0, 0)$ إلى الخلية $(H - 1, W - 1)$ طوله $H + W - 2$.
5. (46 نقطة) لا يوجد أي قيود إضافية.

إذا تحقق أنه في أي من حالات الاختبار كانت استدعاءات الإجرائية `set_instruction` أو برنامج الروبوت خلال تشغيله لا تتوافق مع القيود المشروحة في تفاصيل البرمجة ستنال علامة 0 على كامل المسألة الجزئية

في كل مسألة جزئية يمكنك الحصول على علامة جزئية إذا قام البرنامج بتلوين الخلايا بشكل قريب من الصحيح

بشكل رياضي:

- الحل الخاص بحالة اختبار يكون **كاملا** إذا كان التلوين النهائي للخلايا الفارغة يتوافق مع قواعد مسابقة الروبوت.
- يكون الحل **صحيحاً جزئيا** إذا كان التلوين يبدو كالتالي:
  - يوجد مسار أصغري من $(0, 0)$ إلى $(H - 1, W - 1)$ بحيث يكون لون كل خلية ضمن هذا المسار هو 1 .
  - لا يوجد أي خلية أخرى ضمن المتاهة لونها 1.
  - بعض الخلايا الفارغة ضمن الرقعة لونها ليس 0 وليس 1

إذا لم يكن حلك في حالة اختبار معينة لا صحيح كلياً ولا جزئياً ستكون علامتك على في هذه الحالة 0.

في المسائل الجزئية الأربعة الأولى يمكنك أن تحصل على 100% إذا كان حلك صحيح كلياً و 50% من النقاط إذا كان حلك صحيحاً جزئياً.

في المسألة الجزئية الخامسة تعتمد نتيجتك على عدد الألوان المستخدمة في برنامج الروبوت. بشكل أدق لنرمز لأكبر قيمة لـ Z من أجل كل الاستدعاءات للتابع `set_instruction` بالرمز $Z^\star$ سيتم احتساب نتيجة حالة الاختبار وفقاً للجدول التالي:

الشرط	النتيجة (صحيح كلياً)	النتيجة (صحيح جزئياً)
$11 \leq Z^\star \leq 19$	$20 + (19 - Z^\star)$	$12 + (19 - Z^\star)$
$Z^\star = 10$	31	23
$Z^\star = 9$	34	26
$Z^\star = 8$	38	29
$Z^\star = 7$	42	32
$Z^\star \leq 6$	46	36

علامة كل مسألة جزئية هي علامة أقل حالة اختبار في هذه المسألة الجزئية

Sample Grader

The sample grader reads the input in the following format:

- line 1: H W
- line $2 + r$ ($0 \le r < H$): $m[r][0]$ $m[r][1]$ $\ldots$ $m[r][W-1]$

Here, m is an array of H arrays of W integers, describing the non-boundary cells of the maze. $m[r][c] = 0$ if cell (r, c) is an empty cell and $m[r][c] = 1$ if cell (r, c) is an obstacle cell.

The sample grader first calls `program_pulibot()`. If the sample grader detects a protocol violation, the sample grader prints `Protocol Violation:` `<MSG>` and terminates, where `<MSG>` is one of the following error messages:

- `Invalid array`: $-2 \le S[i] \le Z_{MAX}$ is not met for some i or the length of S is not 5.
- `Invalid color`: $0 \le Z \le Z_{MAX}$ is not met.
- `Invalid action`: character A is not one of `H`, `W`, `S`, `E`, `N` or `T`.
- `Same state array`: `set_instruction` was called with the same array S at least twice.

Otherwise, when `program_pulibot` completes, the sample grader executes Pulibot's program in the maze described by the input.

The sample grader produces two outputs.

First, the sample grader writes a log of Pulibot's actions to the file `robot.bin` in the working directory. This file serves as the input of the visualization tool described in the following section.

Second, if Pulibot's program does not terminate successfully, the sample grader prints one of the following error messages:

- `Unexpected state`: Pulibot recognized a state array which `set_instruction` was not called with.
- `Invalid move`: performing an action resulted in Pulibot moving to a nonempty cell.
- `Too many steps`: Pulibot performed $500\,000$ steps without terminating its program.

Otherwise, let $e[r][c]$ be the state of cell (r, c) after Pulibot's program terminates. The sample grader prints H lines in the following format:

- Line $1 + r$ ($0 \le r < H$): $e[r][0]$ $e[r][1]$ $\ldots$ $e[r][W-1]$

Display Tool

The attachment package for this task contains a file named `display.py`. When invoked, this Python script displays Pulibot's actions in the maze described by the input of the sample grader. For this, the binary file `robot.bin` must be present in the working directory.

To invoke the script, execute the following command.

```
python3 display.py
```

:A simple graphical interface shows up. The main features are as follows

- You can observe the status of the full maze. The current location of Pulibot is highlighted by .a rectangle
- You can browse through the steps of Pulibot by clicking the arrow buttons or pressing their .hotkeys. You can also jump to a specific step
- The upcoming step in Pulibot's program is shown at the bottom. It shows the current state array and the instruction it will perform. After the final step, it shows either one of the error .messages of the grader, or `Terminated` if the program successfully terminates
- To each number that represents a color, you can assign a visual background color, as well as a display text. The display text is a short string that shall be written to each cell having the same color. You can assign background colors and display texts in either of the following :ways
  - .Set them in a dialog window after clicking on the `Colors` button
  - .Edit the contents of the `colors.txt` file
- To reload `robot.bin`, use the `Reload` button. It is useful if the contents of `robot.bin` have .changed