جامعة دمشق
كلية الهندسة المعلوماتية
السنة الثالثة – قسم الذكاء الصنعي

وظيفة عملي مبادئ الذكاء الصنعي

مجموعات الطلاب: ٥ طلاب على الأكثر في كل مجموعة. ويمكن أن يكونوا من فئات مختلفة.

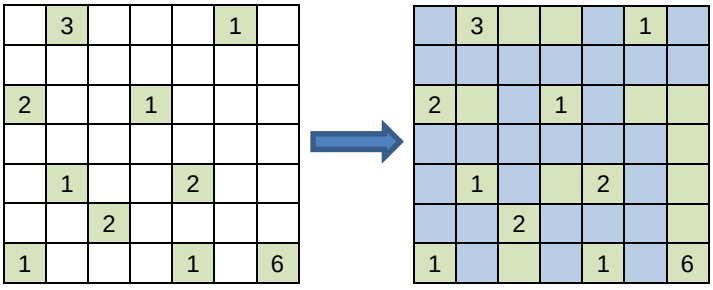
موعد التسليم والمقابلات: يجب تسليم جميع التقارير يوم الثلاثاء ٢٠١٨/٥/٢٩. بينما تتم المقابلات يومي الثلاثاء والأربعاء ابتداءً من الساعة التاسعة صباحاً تكون أفضلية توقيت التسليم بتسلسل أصغر رقم طالب من كل مجموعة، أو أفضلية القدوم بحيث يقابل في اليوم الأول جميع طلاب الفئات الثمانية الأولى ومجموعاتهم حتى لو كان باقي الطلاب من فئات أخرى. لا تمنح علامة للطلاب الذين لا يحضرون المقابلة.

طريقة التقديم: تقرير مطبوع يحتوي الشرح والاستنتاجات والبرامج، إضافة إلى مقابلة مخصصة للعرض والتقييم.

نوريكابي (Nurikabe)

لعبة نوريكابي من ألعاب الأحاجي التي يلعبها لاعب واحد والتي تعتمد على المنطق، سنقوم بكتابة برنامج يقوم بحل هذه اللعبة باستخدام لغة البرمجة برولوغ.

يمكنك تجريب هذه اللعبة على هذا الرابط: http://www.logicgamesonline.com/nurikabe/



تتألف اللعبة من جدول مقسم إلى خلايا، يمكن أن يكون للجدول قياسات مختلفة. بعض هذه الخلايا تحوي على أرقام، هدف اللعبة هو ملء خلايا الجدول الغير حاوية على أرقام إما بخلية زرقاء (تمثل بحر) أو خلية خضراء (تمثل أرض).

يحوي الجدول على جزر بعدد الخلايا المرقمة، بحيث تكون الخلية المرقمة هي إحدى خلايا الجزيرة التي عدد خلاياها هو الرقم داخل الخلية.

يتم ملء الخلايا وفق القواعد التالية:

١. لا يمكن تعديل لون الخلية الحاوية على رقم، ولا تحوي الرقعة على جزر خلاياها ليست إحدى الخلايا المرقمة.
٢. كل جزيرة معزولة عن الجزر الأخرى، أي أن خلاياها لا تلقي عمودياً أو افقياً، ولكن من الممكن أن تتلامس بشكل قطري.
٣. يجب على الخلايا الزرقاء أن تكون متصلة، أي أنه لا يمكن أن تشكل "بحيرة" معزولة ضمن أرض.
٤. لا يمكن وصل الخلايا الزرقاء لتشكل كتلة من 2×2 خلية أو أكثر، بينما يمكن لخلايا الجزر أن تحوي هذه الكتل.

اللعب مع المستخدم (٩ علامات)

أولاً تمثيل المسألة:

*سنقترح طريقة لتمثيل المعطيات، **التزم في حلك بهذه البنية حصراً**، يمكنك الإضافة على هذه البنية ولكن لا تستخدم أي بنية أخرى.*

١- نمثل الخلايا الثابتة ضمن اللوح عن طريق اسناديات، بحيث تعبر عن علاقة بين ثلاث معاملات: السطر والعمود وقيمة الرقم فيها، بحيث يكون لدينا حقيقة من أجل كل خلية ثابتة ضمن الجدول.

$$fxd\_cell(Row, Col, Num)$$

من أجل المثال السابق يكون لدينا ١٠ حقائق على الشكل التالي:

$$fxd\_cell(1,2,3).\ fxd\_cell(1,6,1).\ fxd\_cell(3,1,2)\ldots etc$$

٢- نمثل خلايا الحل بإسناديات (علاقة بين سطر وعمود ولون) مولدة بشكل ديناميكي[١] خلال الحل
من أجل المثال السابق يكون لدينا مثلاً

$$cell(Row, Col, Color)$$

$$cell(1,1, blue).\ cell(1,3, green).\ cell(1,4, green).$$

تذكر أنه يمكنك دائماً تجميع الحلول المختلفة التي تعطيها أي إسنادية (حقائق ثابتة أو قواعد) ضمن سلسلة لاستخدامها لاحقاً باستخدام تعليمة findall[٢]

مساعدة:

[١] تولد الخلايا باستخدام تعليمة assert، ويمكن حذفها باستخدام التعليمات (retract، retractall ...)
لكي تتمكن من التعامل مع أي إسنادية بشكل ديناميكي لابد أن تكون هذه الإسناديات معرفة بأنها dynamic، يتم ذلك بإضافة قاعدة تحوي التعليمة dynamic في بداية ملف الكود على الشكل:

```
:- dynamic t/2.
```

وذلك يعني ذلك أن الإسنادية t التي تأخذ متحولين هي من النوع dynamic

[٢] من المفيد أحياناً تجميع جميع الحلول الممكنة المرتبطة بمتحول ما ضمن قائمة لمعالجتها لاحقاً. يؤمن برولوغ إجرائية لتحقيق هذا الهدف تدعى findall، والتي تكتب على الشكل: findall(X, p(...,X,..), L)، حيث تعيد سلسلة L تحوي على جميع قيم X التي تحقق الإسنادية p. (طبعاً يجب أن تحوي p على المتحول X).
مثال: ليكن لدينا الإسناديات

cell(1,1,blue). cell(1,2,green). cell(1,3,green). cell(1,4,green). cell(1,5,blue). …etc

عندها يكون:

findall(C, cell(_,C,green), L). → L = [2,3,4 …]
findall(c(R,C), cell(R,C,green), L). → L = [c(1,2), c(1,4), c(1,5) …]

ثانياً: قواعد أساسية

بدياً سنكتب مجموعة من القواعد التي تقوم بـ

١. تهيئة اللعبة: والذي يتضمن حذف كل الإسناديات الديناميكية الموجودة مسبقاً (من تشغيل سابق للعبة)[٣]. بالإضافة إلى توليد اسناديات cell تعطي اللون الأخضر لجميع الخلايا الثابتة

٢. طباعة الرقعة: يجب أن تتم هذه العملية بعد كل خطوة لعب، بحيث تطبع الرقعة سطراً سطراً[٤]، بحيث رقماً مكان الخلايا الثابتة وحرف B مكان الأزرق وحرف G مكان الأخضر.

٣. إيجاد مجموعة الخلايا من نفس اللون المتجاورة بالسطر أو العمود والتي تشكل معاً مجموعة (جزيرة أو بحر) ابتداءً من خلية معينة ووضعها في قائمة (مثلاً إيجاد محموعة جوارات الخلية (١,١) في الحل النهائي يعطي قائمة بجميع الخلايا الزرقاء في الرقعة، أما إيجاد جوارات الخلية (1,4) يعطي قائمة بالخلايا (1,2) و (1,3) و (1,4)).
من الممكن أن يساعدك كتابة إجرائيات تقوم بـ (أو قم بتحقيقها على طريقتك)

- إيجاد جميع جوارت خلية بشرط أن يكون له نفس اللون: ابتداءً من خلية لها سطر وعمود ولون يوجد السطر والعمود الذي يعطي جواراً ما للخلية، (الخلية التي فوقها أو تحتها أو على يمينها أو يسارها)، ويتأكد من أن لون الخلية في الحقيقة الديناميكية cell الموافقة له نفس اللون ويضعها في قائمة.
مثلاً جوارت الخلية (1,2) والذي له نفس اللون هو فقط الخلية (1,3)
وجوار الخلية (2,3) والذي له نفس اللون هو الخلايا (2,2) و (2,4) و (3,3)

---

[٣] إن استدعاء الحل أكثر من مرة بدون إعادة تشغيل بروليوغ يؤدي إلى أن القواعد التي تم إضافتها عند التنفيذ السابق لم تحذف، وقد تؤدي إلى عمل البرنامج بشكل خاطئ، يمكنك كتابة قاعدة clear تحذف جميع الحقائق الديناميكية التي قمت بإضافتها باستخدام retractall واستدعائها قبل كل تنفيذ.

[٤] هذه القاعدة (وقواعد اخرى في هذه المسألة) تتطلب المرور على جميع عناصر الجدول، هذا يشبه تطبيق حلقة for على الأسطر والأعمدة، طبعاً هذه التعليمة غير متوفرة في prolog ولكن يمكن كتابة القواعد على الشكل التالي لتقوم بنفس الأثر:

```
% rule for generating numbers from 1 to 7

get_num(Num):- L =[1,2,3,4,5,6,7], member(Num,L).

% rule that needs looping (e.g. print...)

loop_for_some_reason :- get_num(Row), get_num(Col) %predicate that generates
many solutions…
                        process(Row, Col, ..) %do something with one solution
                        ,fail. %so the rule will fire again!
```

كتابة القاعدة بهذا الشكل يؤدي إلى أنها ترد false في كل مرة، بالتالي هذا يدعوها إلى تجريب جميع الاحتمالات الممكنة، ولكن في النهاية هذه القاعدة ترد false دائماً (تفشل دائماً)، بالتالي عند استدعائها في أي قاعدة أخرى يجب أن نقوم بعملية negation (تطبيق not) على هذه القاعدة حتى لا تتسبب بفشل القاعدة التي استدعتها.

```
Solve(… ):-\+loop_for_some_reason, …
```

- لإيجاد جميع جوارات خلية والتي لها نفس اللون يمكنك البدء بوضع الخلية نفسها في قائمة ومن ثم ملء هذه القائمة بالجوارات بشكل عودي (بطريقة top-down أي ابتداءً من أول عنصر في القائمة واحداً واحداً حتى تنتهي القائمة عندها نرد قائمة فارغة)، وتكون الجوارت هي عملية اجتماع بين جميع القوائم للجوارات الناتجة.

ثالثاً: منطق اللعب

لتحقيق منطق اللعب يجب تعريف مجموعة من القواعد تهدف إلى:

١- التحقق من إمكانية إعطاء لون لخلية (أن تكون ليست ثابتة).
٢- التأكد من كون اللعبة قد انتهت: جميع الخلايا لها لون.
٣- التأكد من كون رقعة ما تشكل حل صحيح وذلك بـ:

- التخقق من تحقيق قواعد اللعبة وذلك بإيجاد جميع الجزر والبحور والتأكد من كونها لا تخرق القواعد حيث يكفي أن يخل الحل بقاعدة واحدة حتى يعتبر غير مقبول.
  - البحر وحيد
  - البحر لا يحوي خلايا تشكل كتلة من 2 × 2 خلية
  - الجزر تحوي خلية ثابتة واحدة
  - عدد الخلايا في الجزيرة الواحدة يساوي الرقم في الخلية الثابتة

رابعاً: لعب المستخدم

تبدأ اللعبة بالتهيئة ثم تكرار[٥] مجموعة الخطوات التالية حتى الإنتهاء

١- قراءة دخل من المستخدم: لنتيح للمستخدم إمكانية اختبار حله قبل انتهاء الحل يمكن قراءة محرف من المستخدم بدايةً لنعطي خيار هل يريد ادخل خلية (سطر وعمود ولون) أم اختبار تخقيق قواعد اللعبة على الحل الجزئي الحالي.
٢- إضافة خلية توافق دخل المستخدم.
٣- طباعة الرقعة.
٤- التأكد من كون اللعبة قد انتهت
   - في حال انتهت اللعبة التأكد من صحة الحل
   - في حال لم تنتهي تكرار الخطوات السابقة

[٥] يمكنك نحقيق هذا التكرار بنفس فكرة المرور على الحلقات الموجود في الطباعة.

الكمبيوتر يلعب (تحدي نوريكابي) (٣ علامات)

ليس عليك بالضرورة تحقيق الحل للآخر للحصول على علامة هذا الطلب ولكن في حال قمت بتحقيقة بشكل صحيح ستنال علامات اضافية (أعلى من علامة الوظيفة) لتعويض نقص في علامة تقييماتك

هذه المرة سيقوم الكمبيوتر بحل اللعبة نفسها يمكن تحقيق ذلك **بإحدى طريقتين**

حل المسألة باعتبارها مسألة بحث

يمكن أن ننظر إلى المسألة باعتبارها مسألة بحث تراجعي backtracking ، حيث أننا نبحث عن الحل الوحيد المقبول الذي يملأ الخلايا بشكل صحيح أي يعطي لكل خلية فارغة لون (أرض أو بحر) وذلك ضمن فضاء الحلول الذي يمثل جميع الخيارات الممكنة لملء الخلايا بألوان.

بما أن برولوغ يبحث عن الحلول أصلاً بطريقة التراجعية عن طريق البحث في العمق أولاً (depth first search) فيمكن بسهولة إيجاد الحل، وذلك بتوصيف ما هو الحل المقبول وترك برولوغ يقوم بمسألة البحث.

لحل المسألة بهذا الشكل يمكنك القيام بتوليد حل ما (ملء الرقعة بألوان) وثم التأكد من كون هذا الحل مقبول أم لا،حيث يكفي أن يخل الحل بقاعدة واحدة حتى يعتبر غير مقبول.
في حال كان هذا الحل صحيح يجب طباعة الرقعة بشكل مرتب وواضح.

عادةً تصمم الألعاب بحيث يكون لها حل واحد مقبول، ولكن يجب أن يكون برنامجك قادراً على طباعة جميع الحلول الممكنة.

حل المسألة باستخدام القواعد

على الرغم من أن الطريقة السابقة تعطي حلاً صحيحاً دائماً إلاّ أنه حل مكلف جداً، ولا يأخذ بعين الاعتبار قواعد اللعبة، حيث أن لاعبو نوريكابي يتبعون قواعد تمثل استراتيجيات للحل تأتي من خبرتهم بلعب هذه اللعبة، ويقومون بملء الرقعة تدريجياً بتكرار الحل بهذه القواعد حتى إيجاد الحل كاملاً (في حال كانت هذه القواعد كافية لحل المسألة). كلما كانت قواعدك أكثر تعقيداً كلما كان برنامجك قادراً على حل مسائل أصعب.

قم بحل المسألة بكتابة قواعد تطبق هذه الاستراتيجيات، الروابط التالية تحوي على مجموعة استراتيجيات، حقق ما تستطيع منها:

- http://www.nikoli.co.jp/en/puzzles/nurikabe.html
- http://www.conceptispuzzles.com/index.aspx?uri=puzzle/nurikabe/tutorial
- http://www.conceptispuzzles.com/index.aspx?uri=puzzle/nurikabe/techniques

توجيهات :

- مراعاة ترتيب التقرير، وجودة الحل، وكتابة المتحولات بأسماء معبرة.
- يحوي التقرير على جميع القواعد والإسنادیات لحل المسألة مع **شرح *بسيط وواضح*** لكل منها، بالإضافة إلى **أمثلة اختبار** لتوضيح الفكرة عند الحاجة. مراعاة ترتيب التقرير، وجودة الحل، وكتابة المتحولات بأسماء معبرة.
- ننصح بالتحضير للمقابلة كي تتم خلال ٧ دقائق، يتم خلالها عرض العمل الشخصي بمشاركة جميع طلاب المجموعة والنتائج العملية التي تم التوصل إليها من خلال مثال اختبار مجهز مسبقاً للمسألة كاملة ولكل خطوة جزئية على حدى.
- على جميع طلاب المجموعة المشاركة في حل الوظيفة، والطلاب الذين لم يشاركوا في الحل لن ينالوا أي علامة.
- **أية وظيفة منقولة من جهة ثانية أو منقول منها أو مأخوذة من الإنترنت ستنال علامة الصفر حتماً، نتوخى الالتزام بأمانة الحل.**

مدرسو العملي