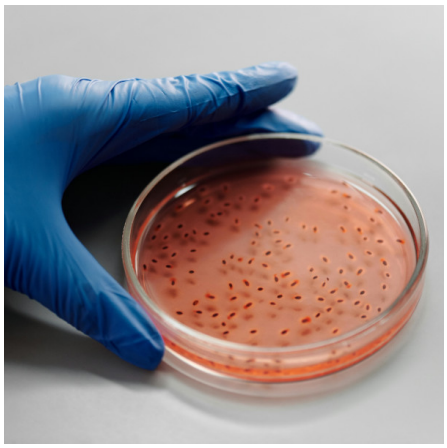
Problema D – Cultura Celular

Este é um **problema de output-only**.
Ao contrário dos outros problemas em que deves fazer leitura de dados e escrita do output, neste problema deves apenas submeter um ficheiro de texto.

Numa placa de Petri representada por uma grelha de dimensão N por M, algumas colónias de células encontram-se inicialmente vivas e outras mortas. As células vivem e reproduzem-se seguindo esta regra iterativa:

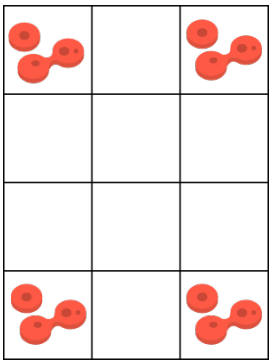


- A cada geração (iteração), uma célula morta torna-se viva se tiver pelo menos duas células ortogonalmente adjacentes já vivas (devido à difusão de nutrientes e sinais bioquímicos). Ou seja, para cada célula morta, contam-se quantos dos seus quatro vizinhos ortogonais (acima, abaixo, à esquerda e à direita) estavam vivos na geração anterior; se pelo menos dois desses vizinhos estiverem vivos, então a célula passa a viva na geração atual. Nota que uma célula viva nunca morre.
- Repete-se até não haver mais células mortas que possam ser ativadas ou até todas as células da placa estarem vivas.

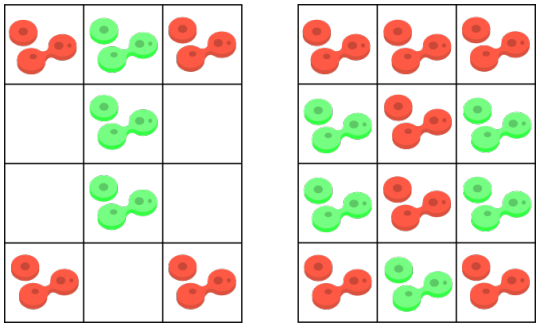
Como biólogo, podes plantar artificialmente até K células adicionais no início (sem remover nenhumas das colónias vivas originais). O objetivo é escolher essas até K células de modo que, ao fim de X gerações, ainda exista pelo menos uma posição sem célula viva, mas que, se deixarmos o processo continuar, todas as células da grelha acabem por ficar vivas.

Exemplo

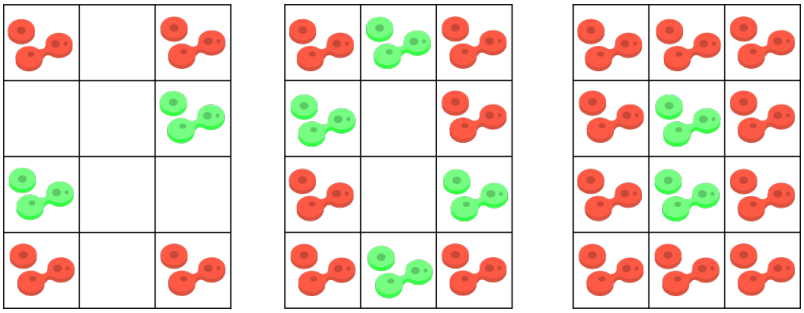
Considera uma placa $N = 3$ por $M = 4$ em que as seguintes células estão vivas:



Supõe que $\boldsymbol{K} = 3$ e $\boldsymbol{X} = 2$. Se plantarmos 3 células extra no centro da grelha, o processo de reprodução leva exatamente duas gerações, como ilustrado na imagem seguinte, onde as células verdes são as células adicionadas na geração atual, ou seja, na primeira imagem correspondem às 3 células adicionadas por nós no início.



Visto que na geração $\boldsymbol{X} = 2$ todas as células estão vivas, esta solução não é válida. Se adicionarmos antes as células seguintes então temos um processo com três gerações:



Visto que na geração $\boldsymbol{X} = 2$ há duas células mortas e na geração 3 todas as células estão vivas, esta solução é válida.

Formato de Input

Cada caso de teste começa com 4 inteiros separados por espaços, N M K X.

Seguem-se N linhas, cada uma com M caracteres, representando o estado inicial da placa de Petri:

- `#` indica uma célula viva,
- `.` indica uma célula morta.

Para o exemplo acima isto seria:

```
3 4 3 2
#.#
...
...
#.#
```

Casos de teste

Há 7 casos de teste, cada um com uma pontuação fixa (sem pontuações parciais para cada um). **Devem olhar para cada caso de teste individualmente, os casos não são arbitrários e devem observar e aproveitar a estrutura de cada um para o resolver**.

Os casos de teste são os seguintes:

- **Caso 1:** A valer 10 pontos, inp1.txt;
  Descrição: $N = M = K = 20, X = 0$.
- **Caso 2:** A valer 15 pontos, inp2.txt;
  Descrição: $N = 15, M = 35, K = 25, X = 0$.
- **Caso 3:** A valer 15 pontos, inp3.txt;
  Descrição: $N = 30, M = 50, K = 38, X = 0$.
- **Caso 4:** A valer 15 pontos, inp4.txt;
  Descrição: $N = 60, M = 80, K = 65, X = 0$.
- **Caso 5:** A valer 15 pontos, inp5.txt;
  Descrição: $N = M = 16, K = 20, X = 100$.
- **Caso 6:** A valer 15 pontos, inp6.txt;
  Descrição: $N = 20, M = 35, K = 40, X = 350$.
- **Caso 7:** A valer 15 pontos, inp7.txt;
  Descrição: $N = 50, M = 80, K = 64, X = 2000$.

Se preferirem, podem fazer download de todos os casos aqui: tests.zip.

Formato de Output/Submissão

Deve ser submetido um único ficheiro de `txt` (o formato é importante, o vosso ficheiro deve terminar em `.txt`). O ficheiro deve conter uma grelha por caso de teste, cada uma com as dimensões do caso de teste a que corresponde, usando `#` para células vivas e `.` para células mortas. Não é permitido remover células vivas originais. Podes apenas plantar até $\boldsymbol{K}$ células adicionais (ou seja, trocar `.` por `#`). O resultado deve satisfazer as condições do enunciado: após $\boldsymbol{X}$ gerações ainda existirá pelo menos uma célula morta, mas se o processo iterativo continuar até ao fim, todas as posições da placa acabarão por ficar com células vivas.

Observações importantes:

- Caso não tenham solução para um dos casos de teste, devem incluir a grelha original.
- Se algum dos outputs for inválido, seja porque não contém caracteres suficientes, ou porque contém caracteres inválidos, o resultado da submissão será `Wrong Answer` e terão 0 pontos em *todos os casos de testes*.

Avaliador de exemplo

Para testar localmente o teu ficheiro de submissão, fornecemos um avaliador de exemplo que segue as mesmas regras descritas no enunciado, aqui: checker.cpp.

Para utilizares este avaliador de exemplo, compila o ficheiro fonte em C (por exemplo, com `g++ -o checker checker.cpp`) e executa-o passando como argumentos o ficheiro de input e o ficheiro de output gerado pela tua submissão. Ou seja: `./checker inpI.txt outI.txt`.

Onde `inpI.txt` deve ser um dos ficheiros de input, e `outI.txt` deve ser o teu ficheiro de submissão correspondente a esse input. O avaliador imprime "Correto" se a tua solução satisfizer todas as condições, ou uma mensagem de erro descrevendo a falha detetada.

Dicas para gerar o output

Visto que é necessário criar um ficheiro de `txt` para a vossa submissão, pode ser útil usar o terminal para gerar os ficheiros de texto (isto não é obrigatório, podem criar o ficheiro como quiserem).

Se para correrem o vosso programa usarem o comando `X` (por exemplo, para `C++` isto seria algo como `./a.out`, depois de compilar com `g++ codigo.cpp`), podem usar o seguinte comando para ler o input de um ficheiro `inp.txt` e escrever para um ficheiro `out.txt`: `X < inp.txt > out.txt`.

Para facilitar a organização do ficheiro de submissão, é aconselhado gerar um `txt` por cada caso de teste, por exemplo, para o caso de teste 2 ter um `out2.txt`.

Para posteriormente gerar um ficheiro `out.txt` para submeter, podem usar o seguinte comando (assumindo que têm um ficheiro `outI.txt` para cada um dos 7 casos de teste):

```
cat out1.txt > out.txt;cat out2.txt >> out.txt;cat out3.txt >> out.txt;cat out4.txt
>> out.txt;cat out5.txt >> out.txt;cat out6.txt >> out.txt;cat out7.txt >> out.txt
```

Organização

APDSI

U.PORTO FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

PRINCETON UNIVERSITY

Alto Patrocínio

Com o Alto Patrocínio de Sua Excelência O Presidente da República

REPÚBLICA PORTUGUESA EDUCAÇÃO, CIÊNCIA E INOVAÇÃO

Patrocinadores

CIÊNCIA VIVA Agência Nacional para a Cultura Científica e Tecnológica

FUNDAÇÃO CALOUSTE GULBENKIAN