E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Introdução à Programação

CÓDIGO: 21173

DOCENTE: José Coelho

A preencher pelo estudante

NOME: Diogo Miguel Palma Sustelo

N.º DE ESTUDANTE: 2201148

CURSO: Licenciatura em Engenharia Informática

TRABALHO / RESOLUÇÃO:

Explicação do programa da alínea A:

- Foi utilizado o vetor “str" para ler a string de entrada, após ser lida a é atribuído o tamanho da mesma ao inteiro “tamanho”.
- Feito este processo é utilizado um ciclo FOR para passar os valores no vetor de char “str” para o vetor de inteiros “tabuleiro”, após o qual é chamado o procedimento “MostraLamberta” que recebe os parâmetros do vetor de inteiros “tabuleiro” e o inteiro “tamanho”.
- Recebidos os valores, usa-os num ciclo FOR com uma cadeia de decisão IF, para atribuir ao vetor de char “tabuleiroox” um caracter “O” caso o valor no vetor de inteiros recebido seja igual a 0, ou em caso contrário atribuir-lhe um caracter “X”.
- De seguida imprime no ecrã o valor do vetor de char “tabuleiroox” mostrando assim o tabuleiro do jogo.
- Além dos testes efetuados pelo VPL, foram feitos testes locais com diferentes inputs, e verificados os respetivos Outputs

Explicação do programa da alínea B:

- Reutilizando o código da alínea A, foi removida o vetor de char “str” e adicionados as variáveis inteiras “saltos” e “tamanho”, que são utilizados para ler a entrada de dois valores distintos.
- Sendo que o valor do inteiro “saltos” é utilizado num ciclo FOR para gerar através da função randaux() o número de valores aleatórios que será para desconsiderar, sendo para considerar apenas o número de valores aleatórios que serão gerados de seguida com outro ciclo FOR idêntico e armazenados no vetor de inteiros “tabuleiro”
- À semelhança da alínea anterior é chamado o procedimento MostraLamberta, com a diferença que os valores recebidos agora são testados da mesma forma, mas para se são pares ou ímpares, atribuindo o caracter “O” aos valores pares e o caracter “X” aos valores ímpares.
- De seguida atribui os caracteres “\0” à última posição do vetor, para terminar a string e imprime no ecrã o valor do vetor de char “tabuleiroox” mostrando assim o tabuleiro do jogo.
- Além dos testes efetuados pelo VPL, foram feitos testes locais com diferentes inputs, e verificados os respetivos Outputs.

Explicação do programa da alínea C:

- Reutilizando o código da alínea B e fazendo as necessárias alterações, adicionadas as variáveis inteiras “jogada”, “casa”, “elementos”,”verificax” e “parajogo” e à semelhança do mesmo, o programa recolhe as variáveis “tamanho” e “saltos” e faz a randomização do tabuleiro e imprime-o, tendo o procedimento “MostraLamberta” sido alterado também para mostrar uma(se só tiver unidades) ou duas linhas(caso tenha dezenas) do numero de casas, através de uma estrutura IF com um ciclo FOR dentro, para as dezenas, caso o valor seja maior que 9, e adicionado ao ciclo FOR que constrói o tabuleiro dos “X” e “O” uma linha que imprime as unidades.
- Foi inicializada a variável “parajogo”, a 0 e adicionado um ciclo FOR, onde testa se o jogo deve continuar, inicializando a variável inteira “jogada” a 1 nesse ciclo, em que o ciclo corre enquanto a variável “parajogo” seja diferente de 1 e é incrementada a variável jogada no fim do ciclo, imprimindo esse ciclo o número da jogada e recolhendo os dados para as variáveis inteiras “casa” e “elementos”, imprimindo-as de seguida com o objetivo do output do programa ser idêntico ao testado pelo VPL.
- Dentro desse ciclo, implementei outro ciclo FOR que testa desde o início da casa inserida até ao fim do segmento pretendido no tabuleiro, se contem “X” (se tem um valor ímpar), atribuindo o valor 1 à variável “verificax”.
- Por último ainda dentro do primeiro ciclo, foi adicionado uma estrutura IF que verifica se o segmento contem “X” (“verificax” ser 1), se o número de elementos não ultrapassa o tamanho do tabuleiro menos a jogada ou se é 1 e se o segmento não ultrapassa o tabuleiro.
- Em caso de ser válido imprime de novo o tabuleiro no ecrã, em caso inválido imprime que a jogada é inválida e que perde o jogador 1 ou 2, conforme o teste da cadeia IF implementada para verificar qual o jogador através de a jogada ser par ou ímpar.
- Além dos testes efetuados pelo VPL, foram feitos testes locais com diferentes inputs, e verificados os respetivos Outputs.

Explicação do programa da alínea D:

- Nesta alínea foi reutilizado o programa da alínea anterior, com a diferença que em vez de receber a jogada através da leitura das variáveis “casa” e “elementos”, essa linha foi substituída por uma chamada ao procedimento “testaregras”, em que se passa o valor das variáveis “tamanho”, “jogada”, “casa”, “elementos” (estes dois últimos passados por referência para quando modificados no procedimento, assumirem o mesmo valor no main) e “tabuleiro”.
- Dentro do procedimento “testax” foram criadas as variáveis locais “contax”,” contabloco” e o vetor de inteiros “conta”.

- O procedimento começa por inicializar as variáveis “contax” e “casa” a 0 e a variável “elementos” a 1.
- É implementado um ciclo FOR que percorre todo o tabuleiro e verifica se existe “X”, existindo incrementa 1 à variável “contax”.
- É implementada uma cadeia de estruturas IF e ELSE encadeados, que começa por testar se a variável “contax” é 0, caso seja, aplica a regra 5, pois mais nenhuma das outras se aplicaria, e imprime no ecrã a regra, já tendo sido inicializadas as variáveis casa e elementos com os valores 0 e 1, não é necessário mais.
- A estrutura continua caso o valor de “contax” seja diferente de 0, indo a cadeia testando sucessivamente e por ordem da 1ª à 4ª regra, caso não se aplique uma, irá passar ao ELSE seguinte, e a partir daqui assume-se que existe pelo menos um “X” no tabuleiro.
- A regra 1 é testada verificando se a jogada máxima é 1, em caso positivo um ciclo FOR procura onde está o último “O” (número par) antes de um “X” e coloca na variável “casa” a posição seguinte, realizando a regra e imprimindo a regra.
- A regra 2 é testada inicializando a variável “contabloco” a 0, com um ciclo FOR é percorrido todo o tabuleiro à procura do primeiro “X”(numero impar),caso encontre incrementa nessa posição(valor de i) do vetor inteiro “conta”, uma unidade, de seguida com outro ciclo FOR inicializado na posição onde se encontrou o 1º “X”, testa igualdades entre essa posição do tabuleiro e a próxima, para definir o tamanho do bloco de “X”, sendo que se encontrar um “X” a seguir a outro, incrementa 1 unidade nessa mesma posição do vetor “conta, enquanto essa condição for verdade e não se encontre na ultima posição do tabuleiro. Ainda na regra 2, de seguida atribui a “j” o valor de “j” menos o valor do tamanho do segmento de “X”(vetor “conta” na posição “i”) mais 1 unidade, para armazenar onde começa a casa a selecionar, e adiciona ao iterador do primeiro ciclo o numero de “X” que encontrou, para assim ir testar o resto do tabuleiro de seguida, tendo também já adicionado ao inteiro “contabloco” 1 unidade, para verificar no fim do primeiro ciclo através de uma estrutura IF se existiu apenas 1 bloco de “X”, e se o tamanho do mesmo não excede o tamanho permitido para a jogada, em caso positivo, realiza a regra e imprime a regra.
- A regra 3 é testada verificando se a jogada máxima é 2 por uma estrutura IF, caso positivo, percorre todo o tabuleiro com um ciclo FOR enquanto na posição correspondente estiver um “O”(numero impar), quando o ciclo parar significa que encontrou um “X”, com uma estrutura IF testa se o numero de “X” é par através da variável “contax” incrementada no inicio do procedimento, se verdadeiro, existe outra estrutura IF dentro desta a testar se é a primeira posição do tabuleiro, caso positivo seleciona essa posição, caso negativo seleciona a anterior, nos dois casos ao numero de elementos é atribuído 2, em caso de o primeiro IF ser

falso, significa que o numero de “X” é impar e à “casa” é atribuído a essa posição, realizando então a regra e imprimindo a regra.

- A regra 4 é testada através de um ciclo FOR que percorre todo o tabuleiro em segmentos iguais ao tamanho máximo permitido na jogada, em que tem outro ciclo FOR dentro que testa nesse segmento quantas alternâncias existe de “X” para “O” e vice versa, através comparação da negação de igualdades de par ou impar, da posição “j” com a “j”+1 e incrementando nas posição correspondente à casa de inicio no vetor “conta” 1 unidade a cada alternância, para de seguida noutro ciclo FOR testar o vetor “conta”, a iniciar no valor de “casa”(definido a 0 no inicio) até ao fim desse vetor que contém o numero de alternâncias a cada posição, caso encontre um valor maior, atribui à variável “casa” o numero dessa posição e no final atribui o numero máximo de elementos à variável “elementos” escolhendo assim o segmento com maior numero de alternâncias, e em caso de empate, seleciona o 1º e com o tamanho máximo possível, perfazendo a regra 4 e imprimindo-a.
- Após o fim do teste das regras, o programa continua como na alínea anterior, tendo havido pequenas alterações nas situações em que existe interação que envolva a variável “casa” pois na alínea C ela assumia valores de 1 até x e agora assume valores de 0 até x.
- Além dos testes efetuados pelo VPL, foram feitos testes locais com diferentes inputs, e verificados os respetivos Outputs, tendo encontrado erros e dificuldades, fui testando e corrigindo à medida que os encontrava.

Nas alíneas A, B e C, não encontrei dificuldades de implementação, tendo os pequenos “bugs” e distrações indo sendo corrigidos à medida que fui implementando e testando os programas

Na alínea D, encontrei alguma dificuldade na regra 2 e regra 3, sendo que na regra 2 a dificuldade encontrada foi como fazer no primeiro ciclo continuar onde termina o 2º ciclo, a fim de testar se existe mais algum bloco de X na continuação do tabuleiro, sem perder os valores necessários para continuar com a regra caso seja o único bloco.

E na regra 3 a dificuldade encontrada, foi em testar e aplicar as regras quando os “X” se encontravam nas pontas do tabuleiro, principalmente se o número de “X” fosse par em que tinha de selecionar 2 elementos válidos, não deteriorando quando a hipótese era distinta, por exemplo selecionar “OX”.

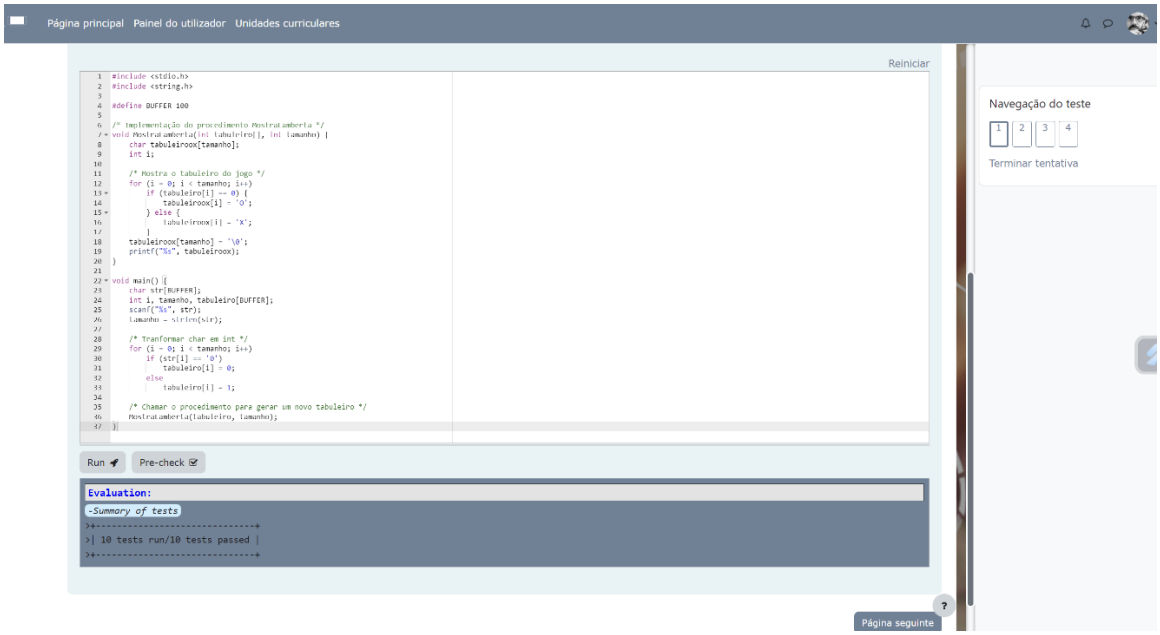
De qualquer das duas dificuldades, após raciocinar mais um pouco e “meter no papel” o que tinha de ser feito, pois às vezes é necessário fazer literalmente “o desenho” para se encontrar uma solução, cheguei ao código descrito nas linhas acima.

Inicialmente implementei as regras das jogadas automáticas no próprio main, mas mais tarde decidi colocar em um procedimento por conta da abstração funcional do programa.

Anexos:

Testes à alínea A:

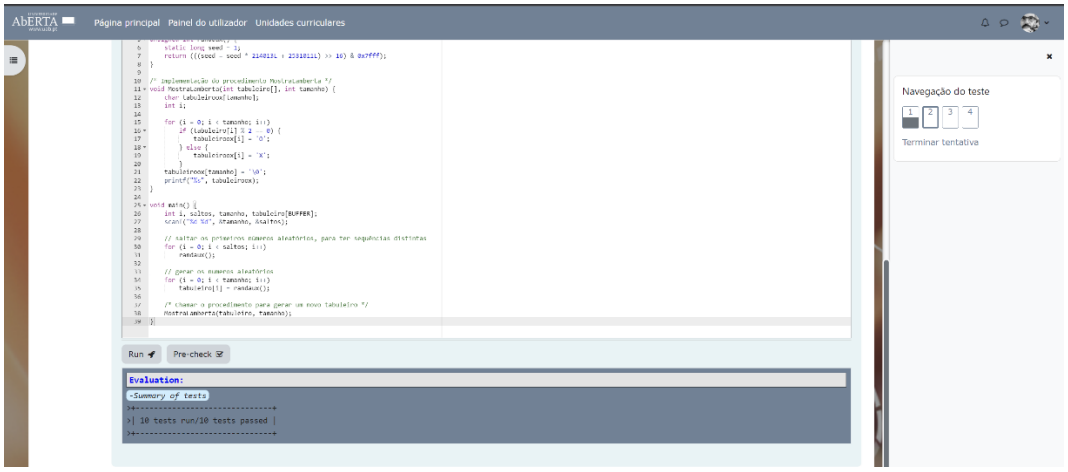
Pre-check do VPL



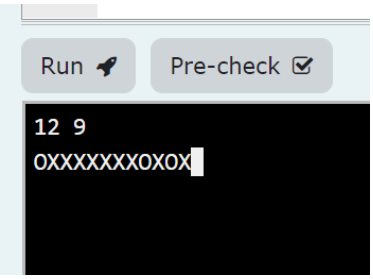
Testes com valores introduzidos manualmente, que comprovam que o programa faz o expectável

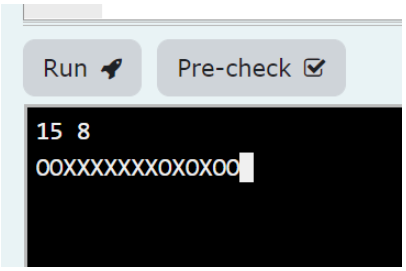
Testes à alínea B:

Pre-check do VPL



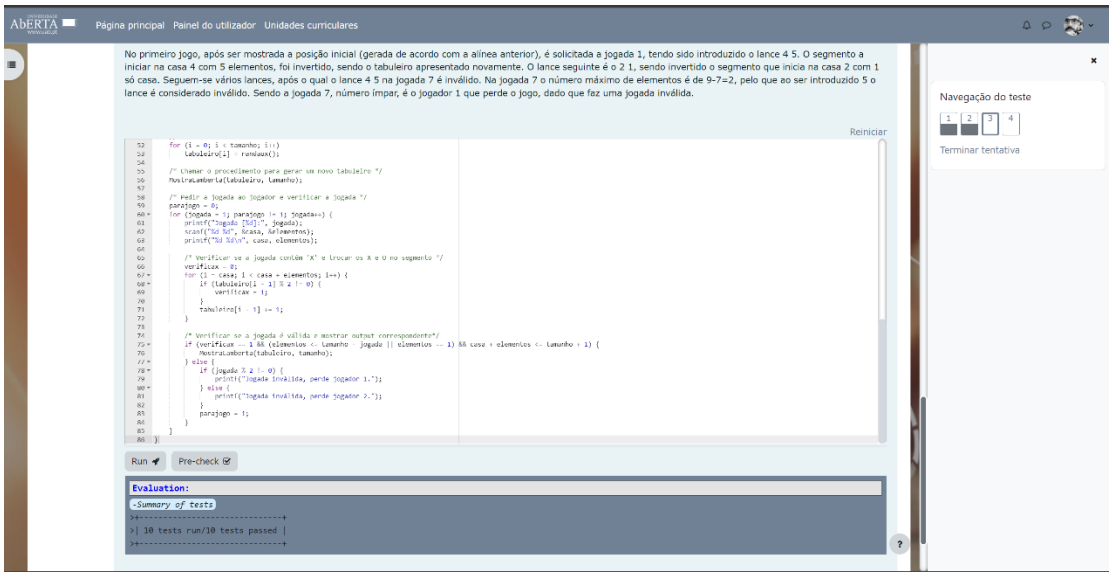
Testes com valores introduzidos manualmente, que comprovam que o programa faz o expectável





Testes à alínea C:

Pre-check do VPL



Testes com valores introduzidos manualmente, que comprovam que o programa faz o expectável

```
5 4
12345
XOOOO
Jogada [1]:1 1
1 1
12345
OOOOO
Jogada [2]:3 2
3 2
Jogada inválida, perde jogador 2.

...Program finished with exit code 0
Press ENTER to exit console.
```

Jogando o único X que aparece no tabuleiro, ele aceita a jogada e troca para O, sendo que na segunda jogada não existe mais X possível e escolhendo a posição 3 o jogador 2 perde.

```
7 5
1234567
OOOOOXX
Jogada [1]:5 2
5 2
1234567
OOOOXOX
Jogada [2]:3 2
3 2
Jogada inválida, perde jogador 2.

...Program finished with exit code 0
Press ENTER to exit console.
```

Jogando com a posição 5 que contém um O, mas com duas posições, selecionando assim OX, a jogada é válida e o jogo prossegue, selecionando depois o 3 e 4 que só contém OO, a jogada é inválida e o jogador 2 perde.

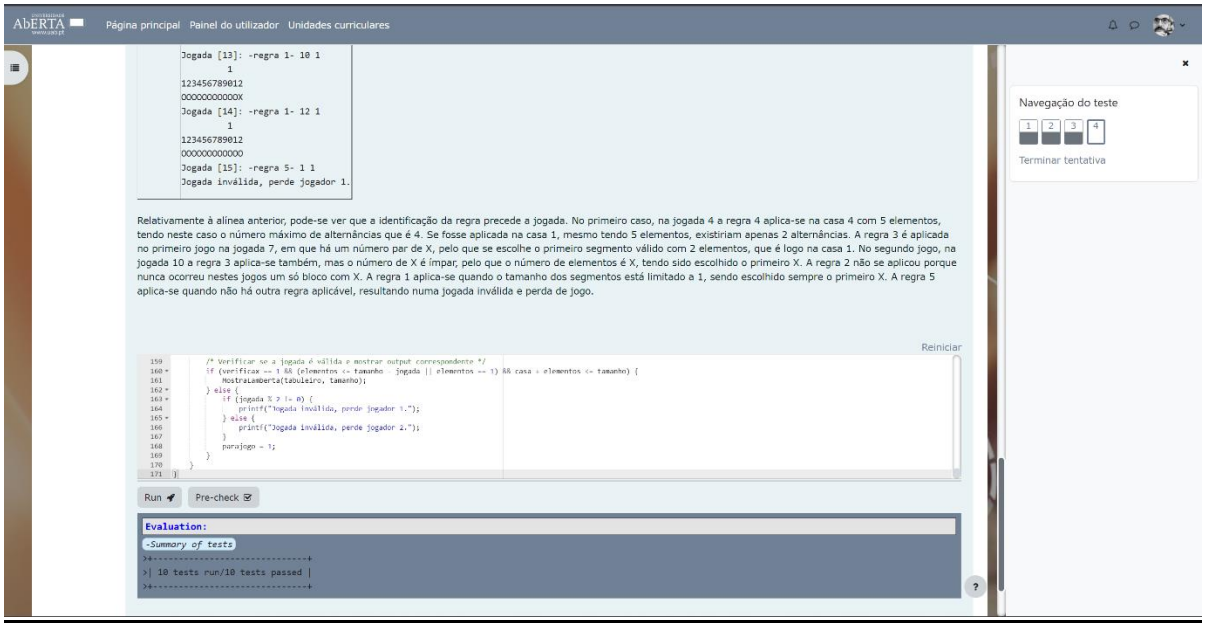
```
3 2
123
OOX
Jogada [1]:1 3
1 3
Jogada inválida, perde jogador 1.

...Program finished with exit code 0
Press ENTER to exit console.
```

Sendo o tabuleiro de tamanho 3 e a jogada 1, o máximo tamanho do segmento poderia ser 2, jogando um segmento de tamanho 3, a jogada é inválida, perde o jogador 1.

Testes à alínea D:

Pre-check do VPL:



Testes com valores introduzidos manualmente, que comprovam que o programa faz o expectável

```
4 15
1234
XXOX
Jogada [1]: -regra 4- 2 3
1234
XOXO
Jogada [2]: -regra 3- 1 2
1234
OXXO
Jogada [3]: -regra 1- 2 1
1234
OOXO
Jogada [4]: -regra 1- 3 1
1234
OOOO
Jogada [5]: -regra 5- 1 1
Jogada inválida, perde jogador 1.

...Program finished with exit code 0
Press ENTER to exit console.
```

Neste caso, consegue ver-se que aplica a regra 4 corretamente, pois da posição 2 a 4 faz 2 alternâncias e da posição 1 a 3 apenas faz 1, aplica também a regra 3 corretamente(como descrito no relatório, foi uma das dificuldades encontradas, quando só podia fazer 2 jogadas, o X encontrava-se numa das pontas, e o numero de X era par, de selecionar a casa correta, tendo sido ultrapassada, como se comprova, consegue também ver-se as regras 1 e 5 a ser feitas corretamente.

```
166         printf("Jogada inválic
167     }
5 9
12345
OXXXX
Jogada [1]: -regra 2- 2 4
12345
OOOOO
Jogada [2]: -regra 5- 1 1
Jogada inválida, perde jogador 2.

...Program finished with exit code 0
Press ENTER to exit console.
```

Neste caso consegue observar-se a aplicação da regra 2, que encontrando apenas um bloco de X continuo e o tamanho dele é 4 e sendo a jogada 1 e tamanho do tabuleiro 5, é valida. Tendo sido esta regra a 2ª dificuldade que encontrei e resolvi com sucesso. No final aplica a regra 5.

```
7 45
1234567
OXXXOXO
Jogada [1]: -regra 4- 1 6
1234567
XOOOXOO
Jogada [2]: -regra 4- 1 5
1234567
OXXXOOO
Jogada [3]: -regra 2- 2 3
1234567
OOOOOOO
Jogada [4]: -regra 5- 1 1
Jogada inválida, perde jogador 2.

...Program finished with exit code 0
Press ENTER to exit console.
```

Mais um teste feito da aplicação regra 4, sendo que é feita enquanto existe alternâncias e sempre escolhendo a maior que exista, e em caso de empate, escolhendo o 1º, quando fica apenas um bloco de X seguido aplica a regra 2 e por fim a regra 5.

```
167 }
XXOXOXO
Jogada [1]: -regra 4- 2 6
1234567
XOXOXOX
Jogada [2]: -regra 4- 1 5
1234567
OXOXOOX
Jogada [3]: -regra 4- 1 4
1234567
XOXOOOX
Jogada [4]: -regra 4- 1 3
1234567
OXOOOOX
Jogada [5]: -regra 3- 1 2
1234567
XOOOOOX
Jogada [6]: -regra 1- 1 1
1234567
OOOOOOX
Jogada [7]: -regra 1- 7 1
1234567
OOOOOOO
Jogada [8]: -regra 5- 1 1
Jogada inválida, perde jogador 2.

...Program finished with exit code 0
Press ENTER to exit console.
```

Um último exemplo de teste efetuado, em que mais uma vez a regra 4 é predominante, encontra novamente a regra 2 numa das pontas(neste caso as duas) mas seleciona a primeira, com 2 casas, pois o número de X é par, continua com a regra 1 e terminando com a regra 5.