Code Xlogo : Jeux de cible

pour jeu :d :a

```
# 3 VARIABLES GLOBALES NÉCESSAIRES POUR LE JEU
donne "depart :d
donne "arrivee :A
donne "pile md :D []
# RÉGLAGE DE L'ENVIRONNEMENT
FSEP 0 VT
donne "consignes (ph [Touche] car 91 "W car 93 [:] anal "fw  [; Touche] car 91 "N car 93 [:] anal "fn [; Touche] car 91 "A car 93 [: annulation dernière commande ; Touche] car 91 "S car 93 [: sortie du jeu.] )
jeu1
```

pour jeu1

```
si egal? :depart :arrivee [FSEP .6 ec [GAGNE! BRAVO...]stop]
ec "-------------------------------------------
ec PH [> Etat de la pile : ] :pile
tape [Position actuelle : ] tape espace 1 FSTY "gras tape :depart FSTY "aucun tape espace 3 tape [Position visée :] tape espace 1 FSTY "gras ec  :arrivee FSTY "aucun
ec :consignes #variable globale définie dans la procédure appelante
donne "suite ecoute [W N A S]
si egal? :suite "S [FSEP .6 ec [PERDU...]stop]
si egal? :suite "A[depile][donne "suite mot "F :suite donne "depart execute ph :suite :depart empile]
jeu1
```

```
fin

pour ecoute :caradmis
donne "clav liscar
si :clav < 0 [ret ecoute :caradmis] [donne "clav majuscule :clav]
si membre? :clav :caradmis [ret :clav][ret ecoute :caradmis]
fin

pour empile
donne "pile md :depart :pile
fin

pour depile
si vide? sd :pile [ec [Annulation interdite !]stop]
ec [Annulation dernière commande]
donne "pile sd :pile donne "depart der :pile
fin

pour majuscule :code
si :code >96 [ret car :code - 32] [ret car :code]
fin

pour espace :n
si :n = 0 [ret "] [ret mot car 32 espace :n - 1]
fin

pour fn :x
ret somme :X 5
fin

pour fw :x
ret somme :X 3
fin

pour demo
# redéfinition de la fonction fn
soit "oldfn texte "fn
soit "comfn item (1 + hasard 2) [somme produit]
soit "argfn 1 + hasard 10
soit "txtfn liste [x] (ph "ret :comfn [:X] :argfn)
definis "fn :txtfn
#redéfinition de la fonction fw
soit "oldfw texte "fw
soit "comfw item (1 + hasard 3) [somme difference quotient]
soit "argfw 2 + hasard 10
tantque [egal? :argfn :argfw] [donne "argfw 2 + hasard 10]
soit "txtfw liste [x] (ph "ret :comfw [:X] :argfw)
definis "fw :txtfw
# definition des bornes du jeu
```

```
soit "terre hasard 13 soit "ciel :terre soit  "fois 3 + hasard 5
repete :fois [si egal? 0 hasard 2 [donne "ciel fn :ciel] [donne "ciel fw :ciel]]
ec (ph [fois=] :fois [comfn =] :comfn :argfn [comfw = ] :comfw :argfw  [terre=] :terre [ ciel=]:ciel)
jeu :terre :ciel
definis "fn :oldfn
definis "fw :oldfw
fin
```

pour anal :nproc

```
# extrait du texte de la procédure nproc (fn ou fw) l'info essentielle
# on cherche dans le texte la première ligne commençant par ret
# ceci permet de rajouter des commentaires
# si la fonction mathématique utilisée n'est pas identifiée, on écrit [??? avec]
soit "letexte texte :nproc soit "laligne []
tantque [vide? :laligne] [si vide? :letexte [donne "laligne "errorerror ] [donne "laligne prem :letexte
donne "letexte sp :letexte si et (non membre? "ret :laligne) (non membre? "retourne :laligne)
[donne "laligne []]] ]
si egal? "errorerror :laligne [ec (ph [gros problème avec la procédure] :nproc [. Vous devriez
recharger le module logo Jeucible.lgo]) stoptout]
si membre? "retourne :laligne [soit "quete "retourne] [soit "quete "ret]
donne "laligne sp membre :quete :laligne
si vide? :laligne  [ec (ph [gros problème avec la procédure] :nproc [. Vous devriez recharger le
module logo Jeucible.lgo]) stoptout]
soit "opé prem :laligne donne "laligne sp :laligne
soit "lacom [???]
si egal? :opé "somme [donne "lacom [ajout de]]
si ou egal? :opé "diff egal? :opé "difference [donne "lacom [retrait de]]
si egal? :opé "produit [donne "lacom [produit par]]
si egal? :opé "quotient [donne "lacom [quotient d'avec]]
soit "largument "
tantque [vide? :largument] [si vide? :laligne [donne "largument "errorerror] [donne "largument
prem :laligne donne "laligne sp :laligne si non nombre? :largument [donne "largument "]] ]
si egal? "errorerror :largument [ec (ph [gros problème avec la procédure] :nproc [. Vous devriez
recharger le module logo Jeucible.lgo]) stoptout]
donne "lacom md :largument :lacom
retourne :lacom
fin
```

pour testxplor

```
# redéfinition de la fonction fn
soit "oldfn texte "fn
soit "txtfn liste [x] (ph "ret "somme [:X 2])
definis "fn :txtfn
#redéfinition de la fonction fw
soit "oldfw texte "fw
soit "txtfw liste [x] (ph "ret "produit [:X 2])
definis "fw :txtfw
fsep 0
```

```
explore "b1 "b3 "32
definis "fn :oldfn
definis "fw :oldfw
fin
```

pour explore :actuelle :cible :valeur

```
# calcule tous les chemins valides pour passer sur un quadrillage de nc colonnes par nl lignes
# d'une position :actuelle à une position :cible et calcule l'évolution de :valeur sous les opérateurs fn et fw
# Le travail se fait pour nc = nl = 3 mais on peut changer les valeurs ci-dessous
donne "nc 3 # variable globale, à modifier sans trop d'excès
donne "nl 3 # variable globale, à modifier sans trop d'excès
# on en déduit les étiquettes de colonne :
donne "nc reste :nc 24 # pas plus de 24 colonnes
donne "derniereColonne item :nc "abcdefghijklmnopqrstuvwxyz
donne "pile []
# on empile :actuelle :valeur :direction
donne "pile md :actuelle :pile
donne "pile md :valeur :pile
donne "pile md "H :pile
# on définit les fonctions inverses des opérateurs [fn] et [fw]
defnb
defwb
vt ec (ph "explore :actuelle :cible :valeur)
# on peut lancer la recherche sur le graphe
fouillegraphe :actuelle :cible :valeur
ec [---------- Terminé]
fin
```

pour fouillegraphe :actuelle :cible :valeur

```
# c'est très récursif, dans 4 directions
si egal? :actuelle :cible [ec affsol :pile :valeur  donne "pile md "nil sd :pile soit "newdir "nil   ]
soit "eticoln prem :actuelle # c'est une lettre
soit "etilign sp :actuelle  # reste un numéro
#dans quelle direction veut-on aller ?
soit "direction der :pile
soit "newactuelle " # en prévision des impossibilités
# vers le haut ?
si egal? "H :direction [soit "newdir "G si :etilign > 1 [soit "newactuelle mot :eticoln difference :etilign 1 soit "newvaleur fn :valeur ] ]
# vers la gauche ?
si egal? "G :direction [soit "newdir "B si precede? "a :eticoln [soit "newactuelle mot car difference unicode :eticoln 1 :etilign soit "newvaleur fwb :valeur ]]
# vers le bas ?
si egal? "B :direction [soit "newdir "D si :etilign < :nl [soit "newactuelle mot :eticoln somme :etilign 1 soit "newvaleur fnb :valeur ]]
# vers la  droite ?
si egal? "D :direction [soit "newdir "nil  si precede? :eticoln  :dernieretColonne [soit "newactuelle
```

```
mot car somme unicode :eticoln 1 :etilign soit "newvaleur fw :valeur]]
# on remplace le haut de la pile pour la prochaine fois
donne "pile md :newdir sd :pile
# si :newvaleur n'est pas nulle, on va pouvoir y aller ce qui suppose au paravant d'empiler ce qu'il faut
si et (non vide? :newactuelle) (non membre? :newactuelle :pile) [donne "pile md :newactuelle :pile donne "pile md :newvaleur :pile donne "pile md  "H :pile fouillegraphe :newactuelle :cible :newvaleur]
# on retourne à la case d'où l'on vient a priori, donc on dépile
si vide? :pile [stop]
soit "newdir der :pile
tantque [et (egal? :newdir "nil) (non vide? :pile)] [
donne "pile sd :pile donne "pile sd :pile donne "pile sd :pile
si non vide? :pile [soit "newdir der :pile]] # on a visité toutes les branches depuis un noeud on dépile
si non vide? :pile [soit "newactuelle der sd sd :pile soit "newvaleur der sd :pile fouillegraphe :newactuelle :cible :newvaleur]
fin
```

pour affsol :lapile :valeur

```
soit "chemin [Cases traversées : ]
tantque [non vide? :lapile] [
donne "chemin md prem :lapile :chemin
donne "lapile sp :lapile
donne "chemin md (mot car 40 prem :lapile car 41) :chemin
donne "lapile sp :lapile
donne "lapile sp :lapile]
donne "chemin (ph :chemin  [Valeur finale = ] :valeur)
ret :chemin
fin
```

pour defnb

```
# redéfinition de la fonction fnb inverse de la fonction fn, à peu de choses près
soit "oldfn der texte "fn
si membre? "somme :oldfn [donne "newfn "difference]
si membre? "difference :oldfn [donne "newfn "somme]
si membre? "produit :oldfn [donne "newfn "quotient]
si ou membre? "quotient :oldfn membre? "div :oldfn [donne "newfn "produit]
soit "argfn der der :oldfn
soit "txtfnb liste [x] (ph "ret :newfn [:X] :argfn)
definis "fnb :txtfnb
fin
```

pour defwb

```
# redéfinition de la fonction fwb inverse de la fonction fw, à peu de choses près
soit "oldfw der texte "fw
si membre? "somme :oldfw [donne "newfw "difference]
si membre? "difference :oldfw [donne "newfw "somme]
```

```
si membre? "produit :oldfw [donne "newfw "quotient]
si ou membre? "quotient :oldfw membre? "div :oldfw [donne "newfw "produit]
soit "argfw der der :oldfw
soit "txtfwb liste [x] (ph "ret :newfw [:X] :argfw)
definis "fwb :txtfwb
fin
```

pour fnb :x

```
ret difference :X 2
fin
```

pour fwb :x

```
ret quotient :X 2
fin
```