Guido van Robot : exercices sans machine

1 Exercice n°1 : Guido à Rolland-Garros

1) Guido veut travailleur à Rolland-Garros comme ramasseur de balles. Le responsable lui fait passer un test : partant de la situation de la figure 1, Guido doit ramasser toutes les balles (les sonnettes) disséminées sur le terrain et les remettre dans la corbeille dans le coin inférieur gauche, pour arriver à la situation de la figure 2.

   Écrivez un code permettant à Guido de réaliser cette opération. Il devra comporter une boucle `do` répétant 6 fois le ramassage d'une rangée de balles, cette opération étant elle-même constituée d'une boucle *do*.

2) Ça y est, Guido est engagé ! Lors de son premier travail, il doit ramasser les balles laissées sur le terrain après l'entraînement d'un champion. Mais il se rend compte que la situation (celle de la figure 3) est un peu différente de celle de son test d'embauche : il y a parfois plusieurs balles au même endroit, et d'autres endroits ne comportent pas de balle.

   Modifiez le code de votre premier programme de manière à ce que Guido puisse accomplir cette nouvelle tâche.

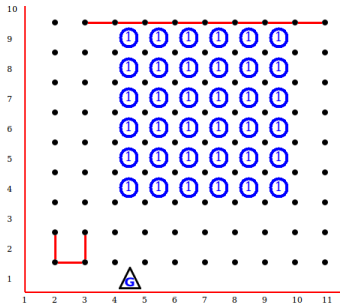

Figure 1

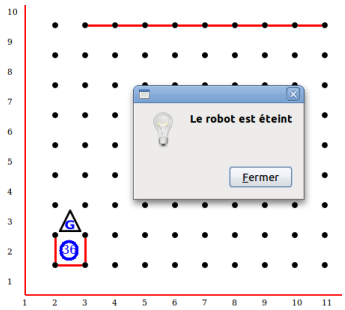

Figure 2

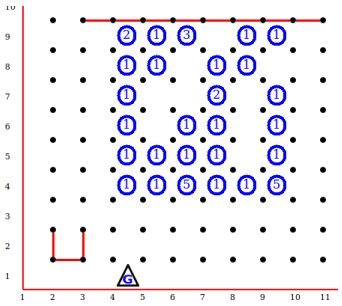

Figure 3

2 Exercice n°2 : Guido fait la course

1) Guido a décidé de faire du sport : il veut se mettre à la course de haie. Une course se passe de la manière suivante : Guido part du coin inférieur gauche, comme indiqué à la figure 4. Il doit courir vers la droite et contourner les haies représentées par des murs, et s'arrêter à la 17ème intersection (figure 5).

   (a) Écrivez une fonction `saute_haie` indiquant à Guido comment "sauter une haie" : partant de la gauche de la haie, regardant vers l'est, il doit se retrouver derrière la haie, regardant encore vers l'est.

   (b) Utilisez cette fonction pour écrire un programme réalisant l'intégralité de la course.

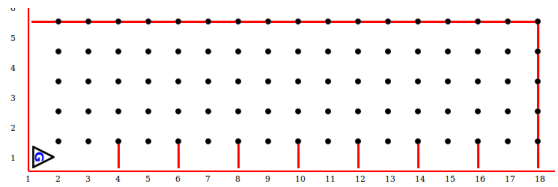

Figure 4

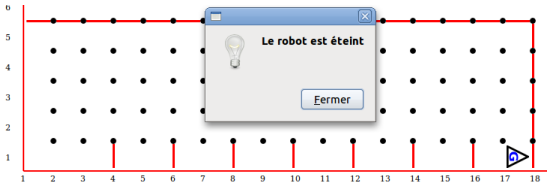

Figure 5

2) Guido trouve la course de haies trop facile, il veut se mettre à la "course de murs" : les obstacles sont disposés aux mêmes endroits sur la piste, mais il sont de hauteur variable (voir figure 6).

   Modifiez votre code de manière à permettre à Guido d'exceller dans ce nouvel exercice.

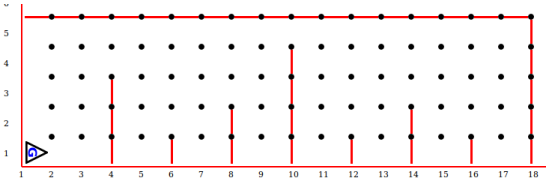

Figure 6

3 Rappel des commandes Guido

Les commandes permettant de contrôler Guido sont de 3 types :

- les commandes *primitives* :
  - `move` : avancer d'une case vers l'avant,
  - `turnleft` : tourner de 90° vers la gauche,
  - `pickbeeper` : ramasser une sonnette,
  - `putbeeper` : déposer une sonnette,
  - `turnoff` : s'éteindre ;
- les *tests* :
  - `front_is_clear` : vrai si il n'y a pas de mur devant,
  - `front_is_blocked` : vrai si il y a un mur devant,
  - `left_is_clear` : vrai si il n'y a pas de mur à gauche,
  - `left_is_blocked` : vrai si il y a un mur à gauche,
  - `right_is_clear` : vrai si il n'y a pas de mur à droite,
  - `right_is_blocked` : vrai si il y a un mur à droite,
  - `next_to_a_beeper` : vrai si la case comporte une sonnette,
  - `not_next_to_a_beeper` : vrai si la case ne comporte pas de sonnette,
  - `any_beepers_in_beeper_bag` : vrai si Guido transporte au moins une sonnette,
  - `no_beepers_in_beeper_bag` : vrai si Guido ne transporte aucune sonnette,
  - `facing_north` : vrai si Guido regarde vers le nord,
  - `not_facing_north` : vrai si Guido ne regarde pas vers le nord,
  - `facing_south` : vrai si Guido regarde vers le sud,
  - `not_facing_south` : vrai si Guido ne regarde pas vers le sud,
  - `facing_east` : vrai si Guido regarde vers l'est,
  - `not_facing_east` : vrai si Guido ne regarde pas vers l'est,
  - `facing_west` : vrai si Guido regarde vers l'ouest,
  - `not_facing_west` : vrai si Guido ne regarde pas vers l'ouest ;
- les *structure* :
  - la *conditionnelle* : "`if <condition>: <actions>`" qui exécute les actions si la condition est vraie, et rien sinon,
  - la *boucle itérative* : "`do <nombre>: <actions>`" qui exécute les actions le nombre de fois indiqué,
  - la *boucle conditionnelle* : "`while <condition>: <actions>`" qui exécute les actions *tant que* la condition reste vraie,
  - la *définition de fonction* : "`define <fonction>: <instructions>`" qui définit `fonction` comme un raccourci pour la séquence des `instructions`.