Misassac

(mini Prologin)

Jean-Baptiste Rouquier

1 Contrat de travail

Vous venez d'être embauché par l'entreprise Moyens Informatiques Super Attractifs de Service aux Salariés Actionnaires et Clients (non, ça ne veut rien dire).

Le management est un peu particulier. Vous disposez de 1000h dans l'année sur votre poste de travail. Comme la maintenance des postes de travail est sous-traitée, toute tentative de travailler plus longtemps est enregistrée comme une tentative de triche.

La gestion se fait par projet. Les projets arrivent un par un, avec une importance (aléatoire) mesurée sur une échelle de 1 à 7. À chaque nouveau projet, chaque employé choisit librement le nombre d'heures qu'il consacre à ce projet, avec un minimum d'une heure. L'employé le plus zèlé reçoit une prime en Nouvelles Unités de Transfert égale à l'importance du projet (aucune prime n'est distribuée en cas d'égalité). Bien sûr, cette prime ne vient pas de nulle part : le salarié qui a travaillé le moins se voit retrancher sur son salaire un nombre de nuts égal à l'importance du projet (c'est le cas pour tous les salariés ayant travaillé le moins en cas d'égalité, et même si la prime n'est pas attribuée).

Enfin, si un salarié a une prime cumulée de -10, il est viré ; si un salarié a une prime cumulée de 10 nuts avant la fin de l'année, il est promu, c'est-à-dire qu'il a gagné la partie (qui s'arrête immédiatement).

Vous devez écrire le code qui choisit le nombre d'heures à consacrer à chaque projet. Cela consiste en trois fonctions :

- `nouvelle_partie : int -> int -> int -> unit`.
  `nouvelle_partie identifiant nb_employes numero_de_la_partie`
  est appelée au début de chaque nouvelle partie. `identifiant` permet d'identifier un employé, par exemple pour se reconnaître dans les listes des fonctions suivantes.
- `joue : int -> (int*etat) list -> int` reçoit l'importance du projet, la liste des couples (identifiant, état) de chaque employé, et renvoie le nombre d'heures que l'employé choisit de consacrer à ce projet.
- `acquiesce : (int*int) list -> unit` reçoit la liste des couples (identifiant, nombre d'heures consacrées à ce projet) de chaque employé. Elle est appelée après chaque projet.

Consultez le fichier types.ml, le fichier main_manual.ml qui vous permet de tester votre code, et les exemples de robots :

- cinquante joue toujours 50 heures ;
- importance_x10 joue 10 fois l'importance

– moyenne joue la moyenne des nombres d'heures jouées sur le projet précédent.

Vous pouvez modifier vos robots autant de fois que vous le souhaitez. Nommez votre fichier prenom_nom.ml, en n'utilisant que des minuscules non accentuées et des soulignés «_» (mais pas deux à la suite). Un classement est établi toutes les 20 minutes avec le dernier robot reçu de chacun, après avoir effectué autant de parties (d'années) que le permet la puissance de calcul à disposition. On affiche alors les résultats, indiquant pour chaque robot : le nombre de parties gagnées, le nombre d'années à la fin desquelles il était encore en train de travailler (il n'avait pas utilisé toutes ses heures quand il y a eu un gagnant), le nombre de tentatives de triche, le nombre d'années où il a épuisé son quota d'heures, et le nombre de fois où il a été licencié. Le but est de gagner le plus de parties, en cas d'égalité d'avoir le moins triché, et s'il y a encore égalité d'avoir été licencié le moins de fois.

2 Aperçu de la moulinette

- Initialisation du générateur aléatoire (pour que l'on ne puisse pas prévoir la suite des nombres aléatoires utilisés...) ;
- mélange de l'ordre des employés (pour que l'on ne puisse pas savoir que l'employé numéro 0 est celui de son voisin) ;
- puis pour chaque partie :
  - appel de la fonction `nouvelle_partie` de chaque employé.
  - et tant qu'il y a des employés en jeu :
    - on tire au sort l'importance du nouveau projet ;
    - on demande à chaque employé combien de temps il y consacre (`joue`) ;
    - on décompte les heures travaillées et on vire les tricheurs (nombre d'heures inférieur ou égal à 0, supérieur au nombre d'heures restantes...) ;
    - on informe les employés valides de ce qui vient d'être joué sans tricher (`acquiesce`) ;
    - on cherche celui qui a travaillé le plus et ceux qui ont travaillé le moins ;
    - on cherche un éventuel gagnant de la partie ;
    - on retire les employés licenciés et les employés épuisés.
- On fait le total sur toutes les parties et on affiche.