TD 6 : Jeu de Yam's

Le jeu de Yam's (ou Yahtzee) est un jeu de dés dont le but est d'enchaîner les combinaisons à l'aide de cinq dés pour remporter un maximum de points.

Nous ne nous intéressons ici qu'à une version simplifiée du Yams et chercherons à reconnaître les figures suivantes :

– **Brelan** : 3 dés identiques
– **Yams** : 5 dés identiques

Les figures permettent de marquer des points. À chacune de ces figures sont associés des bonus : **10 pour le brelan**, et **60 pour le yams**. À cela, on ajoute **la somme des dés qui composent la figure**. Par exemple, les dés $\{2, 5, 3, 5, 5\}$ permettent de marquer 10 points de bonus (brelan) et $5 + 5 + 5 = 15$ points, soit au total 25 points. Le but des prochains exercices est de commencer l'implantation d'un jeu de yams basique.

Exercice 1 (Échauffement : deux fonctions utilitaires)**.**

(1) Spécifier et donner l'implantation d'une fonction `afficheDes` qui affiche le contenu d'un tableau d'entiers en séparant les valeurs par des point-virgules. Ainsi l'appel `afficheDes({1,2,3,5,4})` devra entraîner l'affichage `1;2;3;5;4;`.

(2) Spécifier et donner l'implantation d'une fonction `chercheDansTab` qui cherche l'emplacement d'un entier dans un tableau. Si l'entier est présent dans le tableau, `chercheDansTab` renvoie **l'indice d'une case du tableau le contenant**. Si l'entier n'est pas présent dans le tableau on renverra `-1`. Ainsi l'appel `chercheDansTab(3, {1,2,3})` devra renvoyer `2`, l'appel `chercheDansTab(4, {1,2,3})` devra renvoyer `-1`, etc.

Exercice 2.

(1) Observez les tests suivants et déduisez-en la spécification (rôle, entrées et sortie) de la fonction `compteDes` :

```
ASSERT( compteDes({1,1,1,1,1}) == {5, 0, 0, 0, 0, 0} );
ASSERT( compteDes({2,2,2,2,2}) == {0, 5, 0, 0, 0, 0} );
ASSERT( compteDes({3,3,3,3,3}) == {0, 0, 5, 0, 0, 0} );
ASSERT( compteDes({4,4,4,4,4}) == {0, 0, 0, 5, 0, 0} );
ASSERT( compteDes({5,5,5,5,5}) == {0, 0, 0, 0, 5, 0} );
ASSERT( compteDes({6,6,6,6,6}) == {0, 0, 0, 0, 0, 5} );
ASSERT( compteDes({1,2,3,4,5}) == {1, 1, 1, 1, 1, 0} );
ASSERT( compteDes({2,2,6,2,2}) == {0, 4, 0, 0, 0, 1} );
ASSERT( compteDes({4,1,4,1,1}) == {3, 0, 0, 2, 0, 0} );
```

(2) Proposez une implantation de cette fonction.

Exercice 3 (Yams !)**.**

(1) Le Yams (cinq chiffres identiques) est la figure la plus facile à reconnaître. Donnez la spécification et l'implémentation d'une fonction `pointsFigureYams` qui, lorsque l'on lui donne en entrée un tableau contenant 5 entiers, renvoie les points obtenus (`somme des 5 dés + 60`) s'il s'agit d'un Yams, `0` sinon.

(2) Complétez la liste de tests suivante avec au moins deux autres cas que vous jugez intéressants :

```
ASSERT( pointsFigureYams({4,4,4,4,4}) == 80 );
ASSERT( pointsFigureYams({1,1,1,1,1}) == 65 );
```

(3) Pour simplifier la fonction `pointsFigureYams`, on peut utiliser `compteDes` et `cherCheDansTab`. Donnez une nouvelle implantation de `pointsFigureYams`.

Exercice 4 (Brelan)**.**

(1) À l'image de la question 2 de l'exercice précédent, proposez des tests pour une fonction `pointsFigureBrelan` qui, lorsqu'on lui donne en entrée un tableau contenant 5 entiers, renvoie les points obtenus (`somme des 3 dés qui forment un brelan + 10`) s'il s'agit d'un brelan, `0` sinon.

(2) Spécifiez et implémentez cette fonction en vous aidant de la fonction `compteDes`.

Exercice 5 (Le jeu)**.**

(1) Donner l'implantation d'une fonction `pointsFigure` qui, étant donné un tableau de 5 dés et le nom d'une figure parmi "brelan" et "yams", renvoie le score associé en appelant respectivement la fonction `pointsFigureBrelan` ou la fonction `pointsFigureYams`. Cette fonction doit renvoyer `0` si le nom de figure entré n'est pas valide.

(2) Donner la spécification et l'implantation d'une fonction `lanceDes` qui renvoie un tableau contenant 5 entiers choisis aléatoirement entre 1 et 6. Pour cela vous pouvez utiliser une fonction `int aleaInt(int a, int b)` qui étant donné deux entiers a et b renvoie un entier aléatoire n tel que $a \leq n \leq b$.

(3) Complétez le squelette de code présent dans la **Figure 1** (3 endroits à modifier) pour :
- Lancer les dés
- Afficher le tirage au joueur et lui demander d'entrer une figure **tant que** sa réponse est différente de "brelan", "yams" et "exit"
- Si le joueur choisit "brelan" ou "yams", afficher les points qu'il marque.

(4) Une partie de Yams consiste en de nombreux lancers successifs des dés. Introduisez une boucle supplémentaire pour refléter ce comportement tant que le joueur ne tape pas "exit". De plus faites calculer le score total de la partie, qui est la somme des scores de chaque lancer.

```
string reponseJoueur = "";
vector<int> des;
// INSERER VOTRE CODE ICI

while (     reponseJoueur  != "brelan"
        and reponseJoueur  != "yams"
        and reponseJoueur  != "exit");

  // INSERER VOTRE CODE ICI

  // L'instruction suivante permet d'attendre que le joueur
  // entre une phrase dans le terminal et stocke sa réponse
  // dans une chaine de caractères "reponseJoueur"
  cin >> reponseJoueur;
}
// INSERER VOTRE CODE ICI
```

Figure 1

Exercice ♣ 6 (Relance)**.**
Dans le vrai jeu de Yam's, le joueur peut relancer jusqu'à trois fois un ou plusieurs dés avant de choisir une figure.

(1) Ajouter une fonction `vector<int> relance(int nde, vector<int> des)` qui « relance » uniquement le dé choisi en premier argument et le remplace donc par un nouveau nombre aléatoire entre 1 et 6.

(2) Dans la boucle de jeu, ajouter les instructions nécessaires pour que le joueur puisse choisir jusqu'à trois dés à relancer et les relancer.

Exercice ♣ 7 (Scores)**.**
La partie de Yams se termine lorsqu'un joueur a marqué des points pour toutes les figures possibles.

(1) Ajouter dans le `main` un tableau de score contenant une case pour chaque figure.

(2) Lorsque le joueur choisit une figure, les points qu'il gagne doivent être stockés dans la partie correspondante du tableau. Une fois une case du tableau remplie, elle ne peut plus être modifiée.

(3) La partie se termine lorsque toutes les cases du tableau sont remplies. Le score du joueur correspond à la somme des cases du tableau.