Examen - Première session

Durée : 2h00. ***Documents interdits****. Dans la correction, il sera tenu compte de la clarté du code et de l'utilisation judicieuse de techniques ou conventions telles que les passages par référence, utilisation de* `const`*, séparation entre les méthodes faisant des entrées-sorties et celles n'en faisant pas, bonne gestion de la mémoire et, de façon plus générale, d'une modélisation conforme à la programmation orientée objet en C++14. Afin de gagner du temps, il est inutile d'écrire les accesseurs qui ne sont pas explicitement demandés et dont vous n'avez pas besoin.*

Pour la ***première*** *question, il est demandé de préciser le nom du (des) fichier(s) contenant votre code et donner ces fichiers « en entier », y compris gardes,* `#include`*, etc. Il est inutile de faire un fichier par classe. À partir de la deuxième question, afin gagner du temps, il est inutile de préciser les gardes et vous pouvez (si vous voulez) présenter le code ainsi : écrire à gauche le fichier d'entêtes et, pour chaque méthode déclarée devant être définie, une flèche vers un cadre contenant le code de la méthode (qui devrait habituellement être donné dans le fichier* `.cc` *correspondant).*

1. On veut écrire des classes pour gérer des questions de différents types. Toute question comporte un intitulé, un numéro d'identification (entier identifiant la question, attribué automatiquement) et rapporte un certain nombre de points. Il y a plusieurs types de questions :
   les *SL* (**s**aisie **l**ibre) pour lesquelles on mémorisera la réponse exacte, et qui rapportent 1 point ;
   les *CM* (**c**hoix **m**ultiples) pour lesquelles on mémorisera un ensemble de réponses possibles (vecteur de chaînes) et le numéro (indice dans ce vecteur) de l'unique réponse correcte, et qui rapportent un nombre de points variable, dépendant de la question ;
   les *PR* (**p**lusieurs **r**éponses) pour lesquelles on mémorisera un ensemble de réponses plus ou moins correctes et le nombre de points associé à chacune de ces réponses[1].
   Écrire une (des) classe(s) pour représenter ces différents types de questions ainsi qu'un (des) constructeur(s).

2. Écrire les méthodes `intitule` retournant l'intitulé ; `identifiant` retournant l'identifiant et `nbpoints` retournant le nombre de points maximum pouvant être obtenu à une question.

3. Définir un opérateur de sortie sur flux pour les questions affichant sur ce flux le type de question (SL, CM ou PR), l'identifiant, le nombre de points maximum, l'intitulé, ainsi que, dans le cas d'une question CM, les différents choix précédés de leur numéro[2]. Ainsi, sur une question CM `q`, `std::cout << q;` fera un affichage du type *Question CM - 42 - 1 point. Pour gagner, il faut donner... 1- La bonne réponse 2- La mauvaise réponse.*

4. Écrire une méthode `corriger` prenant comme paramètre une chaîne de caractères, considérée comme une réponse donnée et retournant le nombre de points gagnés :
   Pour une question SL, les points sont gagnés ssi la réponse donnée est la réponse exacte mémorisée ;
   Pour une question CM, les points sont gagnés ssi la réponse donnée est le **numéro** de la réponse correcte mémorisée ou le **texte** (exact) de la réponse correcte mémorisée[3] ;
   Pour une question PR, les points gagnés sont ceux associés à une réponse mémorisée dans le cas où une de ces réponses est donnée, et 0 sinon.

5. Définir une classe `questionnaire` contenant un ensemble de questions représenté par une liste, écrire un constructeur par défaut construisant un questionnaire vide et une méthode `questions` retournant l'ensemble des identifiants des questions d'un `questionnaire`.

1 Par exemple, à la question PR « Quel est l'âge de l'univers en milliards d'années ? », la réponse « 13,8 » pourrait rapporter 3 points, la réponse « 14 » 2 points, et « 13 » 1 point.

2 Les réponses possibles ne sont pas affichées dans le cas d'une réponse PR.

3 Par exemple, sur la question 42 ci-dessus, la méthode retournera 1 (point) si la chaîne passée est « 1 » ou « La bonne réponse », 0 sinon.

6. Écrire une méthode `ajout` permettant d'ajouter une question.

7. Faire en sorte qu'une instance de `questionnaire` puisse être copiée.

8. Écrire une méthode `petitesquestions` prenant comme paramètre un entier `seuil` et retournant toutes les questions dont le nombre de points est inférieur ou égal au `seuil` mais qui ne sont pas des questions CM.

9. Définir une classe `excquest`, sous-classe de `std::exception` que vous définirez et dont le constructeur prendra comme unique argument un identifiant de question et dont la méthode `what()` retournera une chaîne du type « Question introuvable : 43 ».

10. Écrire dans `questionnaire` une méthode `acces` prenant comme paramètre un identifiant de question et retournant la question portant cet identifiant. S'il n'existe pas de question ayant cet identifiant, la méthode lèvera une `excquest`.

11. Écrire une méthode `supprimercm` supprimant toutes les questions CM d'un `questionnaire`.

12. Écrire une classe `ensreponses` mémorisant un ensemble de couples (identifiant de question, réponse donnée).

13. Écrire une méthode `saisie` dans `ensreponses` prenant comme paramètre un `questionnaire` et permettant à un utilisateur de saisir des réponses à chacune des questions : La liste des identifiants de questions du `questionnaire` sera affichée, et l'utilisateur saisira l'identifiant de la question à laquelle il veut répondre, cette question sera affichée (appel à `acces` et opérateur de sortie)[1], et l'utilisateur pourra saisir une réponse qui sera mémorisée dans le `ensreponses`. Un `ensreponses` ne pourra pas contenir deux réponses différentes à une même question.

14. Écrire une méthode `score` qui retourne le nombre de points obtenus par un `ensreponses` à un `questionnaire`.

15. Écrire une classe d'interface graphique avec Qt dont le constructeur recevra un `questionnaire` et un `ensreponses` et qui permettra de faire la même chose que `saisie` :
une *combobox* affichera la liste des identifiants des questions ;
à la sélection d'une question dans ce *combobox* par l'utilisateur, la question sera affichée dans un *label*, et un champ de saisie permettra à l'utilisateur de saisir sa réponse ;
un bouton *Valider* permettra de valider la réponse (c'est-à-dire l'ajouter au `enresponses`) ;
un bouton *Fermer* fermera la boîte et affichera sur la sortie standard le score obtenu.

1 Si lors de l'appel à `acces`, une exception est levée car l'identifiant saisi est introuvable, la méthode affichera « Question introuvable » et demandera de saisir à nouveau un identifiant.