L2

Nom Prénom : .....

2024-2025

Examen Session 2 – Info4A

Exercice (5 points) :

Q1. Comment garantir que les membres d'une classe sont accessibles uniquement par les méthodes de cette classe ?
a) Déclarer les membres comme `protected`.
b) Déclarer les membres comme `private`.
c) Utiliser des fonctions amies.
d) Utiliser des méthodes virtuelles.

Q2. En C++, quel est l'effet de l'utilisation du mot-clé `const` à la fin de la déclaration d'une méthode de classe ?
a) La méthode ne peut modifier aucun membre de la classe.
b) La méthode peut être appelée sur des objets non-const.
c) La méthode doit être redéfinie dans chaque classe dérivée.
d) La méthode ne peut pas être appelée sur des objets const.

Q3. En C++, dans les instructions suivantes, la déclaration de `c2` fait intervenir :

```
C c1;
C c2=c1;
```

a) le constructeur de copie de la classe C
b) la surcharge de l'opérateur '=' de la classe C
c) le destructeur la classe C
d) rien du tout

Q4. En C++, on distingue des méthodes surchargées en fonction :
a) de leurs noms
b) de leurs types de retour
c) du nombre ou du type de leurs paramètres

Q5. Quel est le rôle de l'opérateur de résolution de portée : : en C++ ?
a) Il est utilisé pour accéder aux membres d'une classe depuis ses méthodes.
b) Il est utilisé pour définir des fonctions membres en dehors de la classe.
c) Il est utilisé pour accéder aux membres de l'espace de noms global.
d) Il est utilisé pour accéder aux membres d'une classe de base à partir de classes dérivées.

Q6. Quelle est l'erreur dans le code suivant ?

```
class Carre: public Rectangle {
public:
    Carre(double c) : Rectangle(c, c) {}
};
```

a) La classe `Carre` ne devrait pas être dérivée de `Rectangle`.
b) Les arguments de la classe de base (`Rectangle`) doivent être mis entre crochets.
C) Le constructeur de `Carre` devrait appeler le constructeur de `Rectangle` avec deux arguments distincts.
d) Il n'y a aucune erreur dans le code.

Q7. La surcharge de fonction peut également être obtenue si deux fonctions ou plus ne diffèrent que par leur type de retour.
a) Vrai
b) Faux

Q8. Quand le destructeur est appelé ?
a) Quand un programme se termine
b) Quand une fonction se termine
c) Lorsque l'opérateur « `delete` » est utilisé
d) Toutes les réponses sont vraies

Q9. En C++, il est possible que la classe X, la classe Y et la classe Z héritent de la classe A.
a) Vrai
b) Faux

Examen langage C - session 2

Durée indicative : 1h30

Documents autorisés : 6 feuilles recto verso avec contenu libre.

Calculatrices et objets connectés interdits. Répondez directement sur le sujet, que vous glisserez dans la copie double anonymisée. Écrivez votre numéro d'anonymat sur la première page du sujet. Le barème est donné sur 100 points, avec coefficient 75% (les 25% restants concernent la partie C++).

Partie B

Question 1 (10 points)

Donnez les affichages réalisés par l'exécution des lignes suivantes.

```
printf("%d %d %d ", 7 & 4, 7 | 4, (1 << 4) | 1);
printf("%x %x\n", 0x123 >> 4, 0b10001111);
```

Question 2 (15 points)

Définissez une fonction ...

```
uint8_t exgNibbles(uint8_t data)
{




}
```

... qui retourne le mot binaire obtenu en échangeant les 4 bits les plus à gauche et les 4 bits les plus à droite de l'octet représenté par le paramètre `data`. Par exemple, si cet octet s'écrit `01100011` en binaire, alors la valeur retournée par la fonction doit s'écrire `00110110` en binaire. Vous ne devez pas utiliser de boucle.

Parties C et D

On appelle **miroir** d'une séquence de caractères $c_0, \ldots, c_{k-1}$ la séquence $c_{k-1}, c_{k-2}, \ldots, c_0$. Par exemple, le miroir de `'T', 'i', 'm'` est `'m', 'i', 'T'`.

Question 3 (25 points)

Définissez la fonction ...

```
void miroir(char* s, int k)
{









}
```

...qui remplace la séquence des `k` premiers caractères de la chaîne par son miroir, sans changer le reste de la chaîne. Par exemple, l'exécution du code suivant...

```
char s[] = "Timeo";
miroir(s,3);
printf("%s\n",s);
```

...doit produire l'affichage `miTeo` .

Parties C, D et E

On suppose que la fonction `miroir` de l'exercice précédent est correctement implémentée et disponible.

Question 4 (25 points)

Définissez la fonction...

```
char* dupMiroir(const char* s, int p1, int p2)
{









}
```

... qui crée dans le tas la chaîne contenant le miroir de la séquence de caractères comprise inclusivement entre les positions `p1` et `p2` dans la chaîne désignée par le paramètre `s`. Par exemple, l'appel `dupMiroir("abcdefgh",2,5)` doit retourner l'adresse d'une chaîne `"fedc"`. La fonction `dupMiroir` **doit** appeler la fonction `miroir`.

Partie F

On souhaite représenter un planning d'occupation d'une salle par des personnes. Chaque personne est représentée par une instance de la structure `user` suivante :

```
typedef struct
{
    char id[4]; // identifiant (au plus trois lettres)
    int nc;     // nombre de créneaux occupés
}user;
```

L'attribut `id` représente un identifiant d'utilisateur sous la forme d'une chaîne de caractères d'au plus trois lettres, par exemple "OB". L'attribut `nc` indique le nombre de créneaux occupés par cet utilisateur.

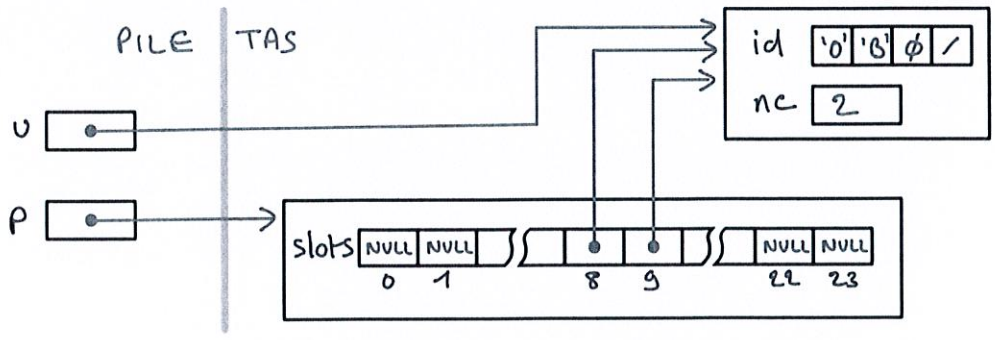
Le planning est représenté par la structure suivante :

```
typedef struct
{
    user* slots[24]; // occupant pour chaque créneau de 1h
}planning;
```

Les lignes de code suivantes permettent de créer un planning dans le tas et de réserver les créneaux 8 et 9 à OB :

```
planning* p = newPlanning();
user* u = newUser("OB");
assign(p, u, 8);
assign(p, u, 9);
```

À l'issue de l'exécution de ces lignes de code, la configuration en mémoire suivante est obtenue :



Il n'est **pas** demandé de définir la fonction `newPlanning`, qui crée dans le tas une instance de `planning` dont le tableau interne est rempli avec des pointeurs `NULL`.

Question 5 (15 points)

Définissez la fonction `newUser`.

Question 6 (10 points)

Définissez la fonction `assign`.
